GENERALIZED VERTEX COLORING PROBLEMS USING SPLIT GRAPHS

THÈSE N^{O} 3629 (2006)

PRÉSENTÉE LE 13 OCTOBRE 2006

À LA FACULTÉ DES SCIENCES DE BASE

Chaire de recherche opérationnelle ROSE

SECTION DE MATHÉMATIQUES

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Tınaz EKİM

DEA de méthodes scientifiques de gestion, Université de Paris IX, France
de nationalité turque

acceptée sur proposition du jury:

Prof. T. Mountford, président du jury
Prof. D. de Werra, directeur de thèse
Prof. M. Demange, rapporteur
Prof. P. Hell, rapporteur
Prof. T. Liebling, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Lausanne, EPFL
2006

Remerciements

Je tiens à remercier chaleureusement plusieurs personnes qui ont contribué à la réalisation de cette thèse :

- mon directeur de thèse, le Professeur Dominique de Werra, pour sa grande disponibilité, ses encouragements constants, la joie de vivre qu'il transmet à son entourage et l'ambiance qu'il a su créer au sein de la ROSE. Je le remercie particulièrement pour m'avoir contaminé du virus de la théorie des graphes que je vais essayer de transmettre à d'autres personnes au long de ma carrière ;
- les membres de mon jury pour avoir aidé à perfectionner la présentation de ma thèse par leurs lectures minutieuses et leurs remarques pertinentes ;
- les relecteurs de ma thèse : Ebrahim pour l'avoir relue en partie ; Murat et Burak sans qui le résumé en turc n'aurait pas vu le jour. Je remercie en particulier Nicolas Zufferey qui était présent pendant deux périodes importantes de ma vie. Au début de ma carrière à la ROSE, il m'a donné toutes les instructions nécessaires et pas nécessaires. A la fin de mon doctorat, il a entièrement relu ma thèse et m'a appris à cette occasion les vertus des virgules ;
- tous mes collègues de la ROSE qui ont contribué à l'ambiance exceptionnelle du groupe : Jocelyne par son énergie positive et son efficacité, Bernard qui écoute toutes mes hésitations sur ma carrière, Benji par ses talents d'organisateur, Telma par sa disponibilité pour papoter, Şivan qui équilibre le poids de Windows contre celui de Linux au sein du groupe, Ivo qui nous donne de l'altitude, David qui arrose bien les soirées, Daniela qui est une excellente collègue, Sacha, Simona et David Cariolaro par leur amitié et les moments partagés ;
- tous mes professeurs qui m'ont fait aimer les Mathématiques et la Recherche Opérationnelle tout au long de mon cursus. En particulier, je suis reconnaissante à Vangelis Paschos de m'avoir fait découvrir et apprécier la théorie de la complexité, à N.V.R. Mahadev pour la période courte mais intense de recherche passée ensemble, et à Marc Demange pour son inépuisable soif de recherche ;
- tous les membres de ma famille qui m'ont soutenu pendant ma thèse et qui ont supporté mon absence ; plus particulièrement ma grande-mère pour être la personne la plus optimiste et la plus encourageante non seulement pendant ma thèse mais aussi pendant toute ma vie. Finalement, évidemment, un grand merci à mon mari, Ahmet, qui est toujours à mes côtés et à mon écoute.

Abstract

Graph theory experienced a remarkable increase of interest among the scientific community during the last decades. The vertex coloring problem (Min Coloring) deserves a particular attention since it has been able to capture a wide variety of applications. For mathematicians, it is interesting for an additional reason: it is extremely hard to solve it in an efficient way.

In this thesis, we introduce several problems generalizing the usual vertex coloring problem, and hence, extending its application domain. We say that a graph is (p, k)-*colorable* if its vertex set can be partitioned into p cliques and k stable sets. Then, for a given p (respectively k), one may ask the following questions: how to choose p cliques (respectively k stable sets) to be removed from the graph such that the number of stable sets (respectively cliques) partitioning the remaining vertices is minimized? These are called (p, k)-*coloring problems*. We also introduce *Min Split-coloring* which is, given a graph G, the problem of minimizing k such that G is (k, k)-colorable. Along the same line, given a graph G, the objective of the problem *Min Cocoloring* is to minimize $p + k$ such that G is (p, k)-colorable. All these problems, called together *generalized coloring problems*, are obviously at least as difficult as Min Coloring. The purpose of this dissertation is to study generalized coloring problems in some restricted classes of graphs in order to bring a new insight on the relative difficulties of these problems. To this end, we detect in a more precise way the limits between $\mathcal{NP}$-hard and polynomially solvable problems.

Chapter 1 introduces generalized coloring problems by emphasizing some preliminary results which will guide the questions to handle in the following chapters.

Chapter 2 exposes the first class of graphs, namely cacti, where Min Split-coloring is shown to be polynomially solvable. We also observe that generalized coloring problems can be polynomially solved in triangulated graphs.

The main result of Chapter 3 is a new characterization of cographs: it is equivalent to say that G is a cograph, and to state that, for every subgraph $G' \subseteq G$, G' is (p, k)-colorable if and only if $G'[V \setminus K]$ is $(p-1, k)$-colorable, where K induces a maximum clique of G'. This result implies simple combinatorial algorithms to solve all generalized coloring problems; the one for Min Cocoloring improves the best time complexity known so far.

In Chapter 4, we handle the recognition of polar graphs which can be seen as a particular (p, k)-coloring, where p cliques are independent (i.e., not linked at all) and k stable sets

form a complete k-partite graph. It is known that the recognition of polar graphs is $\mathcal{NP}$-complete. Here, we determine the first class of graphs, namely cographs, where the polar graphs can be recognized in polynomial time, more precisely in time $\mathcal{O}(n \log n)$. We also give a characterization by forbidden subgraphs. In the same manner, we characterize monopolar cographs, i.e., cographs admitting a polar partition with at most one clique or at most one stable set.

Chapter 5 is devoted to generalized coloring problems in line graphs. Here, we detect the first classes of graphs, namely line graphs of trees, line graphs of bipartite graphs and line graphs of line-perfect graphs, where generalized coloring problems diverge in terms of $\mathcal{NP}$-hardness.

In Chapter 6 we study the approximability of generalized coloring problems in line graphs, in comparability graphs and in general graphs. We derive approximation algorithms with a performance guarantee using both the standard approximation ratio and the differential approximation ratio. We show that both Min Split-coloring and Min Cocoloring are at least as hard as Min Coloring to approximate from the standard approximation ratio point of view, whereas, they admit a polynomial time differential approximation scheme and Min Coloring only a constant differential approximation ratio. We also show that Min Cocoloring reduces to Min Split-coloring in all classes of graphs closed under addition of disjoint cliques and under join of a complete k-partite graph.

In Chapter 7, we handle two different applications of Min Split-coloring in permutation graphs. They give birth to a new problem, called *Min Threshold-coloring*, that we study in the same spirit as the other generalized coloring problems.

In the last chapter, we present several open questions arising from this thesis.

Keywords : Generalized vertex coloring, Split-coloring, Cocoloring, Polar graphs, Approximation

Résumé

La théorie des graphes a eu en essor remarquable dans la communauté scientifique pendant les dernières décennies. Le problème de coloration de sommets (Min Coloration) mérite une attention particulière puisqu'il a de nombreuses applications. Il intéresse les mathématiciens pour une raison de plus : il est extrêmement difficile à résoudre de manière efficace.

Dans cette thèse, nous introduisons plusieurs problèmes qui généralisent le problème usuel de coloration de sommets, et par conséquent, étendent son domaine d'application. Un graphe est (p, k)-*colorable* si son ensemble de sommets peut être partitionné en p cliques et k ensembles stables. Pour un p (respectivement k) donné, nous considérons les problèmes suivants : comment choisir les p cliques (respectivement k ensembles stables) à enlever du graphe de manière à minimiser le nombre d'ensembles stables (respectivement cliques) qui partitionnent les sommets restants ? Ces problèmes sont appelés *les problèmes de* (p, k)-*coloration*. Nous introduisons aussi le problème *Min Coloration Scindée* qui consiste, pour un graphe G donné, à minimiser k tel que G soit (k, k)-colorable. Par ailleurs, pour un graphe G donné, l'objectif du problème *Min Cocoloration* est de minimiser la somme $p+k$ tel que G soit (p, k)-colorable. Tous ces problèmes qui se nomment *problèmes de coloration généralisée*, sont évidemment au moins aussi difficile que Min Coloration. Le but de cette thèse est d'étudier les problèmes de coloration généralisée dans des classes de graphes restreintes afin de pouvoir mettre en évidence les difficultés relatives de ces problèmes. Pour y parvenir, on détermine de façon précise les limites entre les problèmes $\mathcal{NP}$-difficiles et ceux qui sont solubles polynomiallement.

Le premier chapitre présente les problèmes de coloration généralisée en mettant en évidence certains résultats préliminaires qui vont guider les questions considérées dans les chapitres suivants.

Le second chapitre présente la première classe de graphes, à savoir les cactus, où on peut résoudre Min Coloration Scindée en temps polynomial. De plus, nous observons que les problèmes de coloration généralisée peuvent être résolus en temps polynomial dans les graphes triangulés.

Le résultat principal du troisième chapitre est une nouvelle caractérisation des cographes : il est équivalent de dire que G est un cographe et d'affirmer que, pour tout sous-graphe $G' \subseteq G$, G' est (p, k)-colorable si et seulement si $G'[V \setminus K]$ est $(p-1, k)$-colorable, où K est une clique maximum de G'. Ce résultat implique des algorithmes combinatoires simples pour

résoudre tous les problèmes de coloration généralisée ; celui proposé pour Min Cocoloration améliore la meilleure complexité de temps connue jusqu'à présent.

Dans le quatrième chapitre, nous considérons la reconnaissance des graphes polaires que l'on peut également voir comme une (p, k)-coloration particulière, où les p cliques forment une union disjointe de cliques (i.e., sans aucune arête entre elles) et les k ensembles stables forment un graphe k-parti complet. Il a déjà été démontré que la reconnaissance des graphes polaires est $\mathcal{NP}$-complète. Nous déterminons la première classe de graphes, à savoir les cographes, où les graphes polaires peuvent être reconnus en temps polynomial, plus précisément en temps $\mathcal{O}(n \log n)$. Nous caractérisons également les cographes polaires par des sous-graphes interdits. De la même manière, nous caractérisons les cographes monopolaires, i.e., les cographes qui admettent une partition polaire avec au plus une clique ou au plus un ensemble stable.

Le cinquième chapitre est consacré aux problèmes de coloration généralisée dans les graphes aux arêtes. Nous déterminons les premières classes de graphes où les problèmes de coloration généralisée divergent en termes de $\mathcal{NP}$-complétude. Ces classes sont les graphes aux arêtes des arbres, les graphes aux arêtes des graphes bipartis et les graphes aux arêtes des graphes arête-parfaits.

Dans le sixième chapitre, nous étudions l'approximabilité des problèmes de coloration généralisée dans les graphes aux arêtes, les graphes de comparabilité et les graphes généraux. Nous proposons des algorithmes d'approximation avec une garantie de performance en utilisant les rapports d'approximation classique et différentiel. Nous montrons que Min Coloration Scindée et Min Cocoloration sont tous les deux au moins aussi difficiles à approcher que Min Coloration du point de vue du rapport d'approximation standard. Par contre, ils admettent un schéma d'approximation différentiel en temps polynomial et Min Coloration admet seulement un rapport d'approximation différentiel constant. Nous montrons également que Min Cocoloration se réduit à Min Coloration Scindée dans toutes les classes de graphes fermées par les opérations d'adjonction de cliques disjointes et de liaison complète avec un graphe k-parti complet.

Dans le septième chapitre, nous considérons deux applications de Min Coloration Scindée dans les graphes de permutations. Elles donnent naissance à un nouveau problème, appelé *Min Coloration avec Seuil*, que nous abordons avec la même méthodologie que celle utilisée pour les autres problèmes de coloration généralisée.

Dans le dernier chapitre, nous présentons plusieurs questions ouvertes qui surgissent de cette thèse.

Mots-clés : Coloration de sommets généralisée, Coloration Scindée, Cocoloration, Graphes polaires, Approximation

Özet

Graf teorisi son yıllarda daha da artan bir ilgi ile gelişimini sürdürmektedir. Tepe boyama problemi, geniş bir uygulama alanına sahip olmasının yanısıra, etkin bir çözümünün bulunmasınının son derece zor olması nedeniyle de bu gelişmede özel bir yere sahiptir.

Bu çalışmada, tepe boyama problemini ve dolayısıyla uygulama alanını genelleştiren yeni problemler incelenmiştir. Tepe kümesi, p tane kliğe ve k tane bağımsız kümeye parçalanabilen bir graf (p,k)-*boyanabilir* olarak adlandırılmıştır. Verilen bir p (ya da k) doğal sayısı için, graftan silinecek p kliği (ya da k bağımsız kümeyi), geriye kalan tepeleri parçalayabilecek bağımsız küme (ya da klik) sayısını en aza indirecek şekilde nasıl seçmeliyiz sorusu gözönüne alınmıştır. Bu çalışmada, ifade edilen soru (p,k)-*boyama problemi* olarak adlandırılmıştır. Bununla birlikte, verilen bir G grafının (k,k)-boyanabilir olacak şekilde k sayısının en küçük değerini bulma problemi *yarık-boyama problemi* başlığı altında tanımlanmıştır. *Tam-boyama problemi* olarak tanımlanan, verilen bir G grafının (p,k)-boyanabilir olmasını sağlayacak en küçük $p+k$ doğal sayısının bulunması problemi de incelenmiştir. Yukarıda ifade edilen problemler *genelleştirilmiş boyama problemleri* başlığı altında toplanırsa, bunların en az "klasik" boyama problemi kadar zor oldukları aşikardır. Bu tezin amacı, genelleştirilmiş boyama problemlerini kısıtlanmış bazı graf sınıflarında inceleyerek, incelendikleri sınıflardan kaynaklanan $\mathcal{NP}$-zor veya polinomyal olmalarına yol açan nedenleri ortaya koymak ve bu şekilde, belirtilen problemlerin birbirlerine göre zorluklarına açıklık getirmektir.

Birinci bölüm, genelleştirilmiş boyama problemlerine ait tanım ve özelliklerle, daha sonraki bölümlerde kullanılacak temel bilgileri içermektedir.

İkinci bölüm, yarık-boyama probleminin polinomyal çözülebilir olduğu graf sınıflarından bilinen ilk sınıf olan kaktüsleri ele almaktadır. Bu bölümde ek olarak, genelleştirilmiş boyama problemlerinin üçgensel graflar için polinomyal çözülebilir olduğu gözlemlenmektedir.

Üçüncü bölümde elde edilen en önemli sonuç kografların yeni bir karakterizasyonudur. Buna göre, bir G grafının kograf olması eşdeğer biçimde şu şekilde ifade edilebilir : her bir $G' \subseteq G$ altgrafı için, G''in bir (p,k)-boyanabilir graf olabilmesi için gerek ve yeter koşul, K kliği G' grafının bir maksimum kliği olmak üzere $G'[V \setminus K]$'nın $(p-1,k)$-boyanabilir olmasıdır. Bu sonuç yardımıyla genelleştirilmiş boyama problemlerini çözen kombinatoryal algoritmalar sunulmuştur. Buna göre, tam-boyama problemi için literatürdeki algoritmalarla kıyaslandığında en iyi zaman karmaşıklığı elde edilmiştir.

Dördüncü bölümde, p kliğin ayrık olduğu (aralarında hiçbir ayrıt bulunmadığı) ve k bağım-

sız kümenin tam k-parçalı bir graf oluşturduğu (aralarında mümkün olan tüm ayrıtların bulunduğu) özel bir (p,k)-boyama problemi olarak da düşünülebilecek olan, bir grafın kutupsal olup olmadığını belirleme (tanıma) problemi ele alınmıştır. Kutupsal grafların tanınmasının $\mathcal{NP}$-tam olduğu bilinmektedir. Kutupsal grafların kograflar içinde $\mathcal{O}(n \log n)$ zamanda tanındığı gösterilerek, kutupsal grafların polinomyal zamanda tanındığı bilinen ilk kısıtlı graf sınıfı bulunmuştur. Bunun yanı sıra, yasak altgraflar ile kutupsal grafların karakterizasyonu verilmiştir. Benzer şekilde, en fazla bir klik ya da en fazla bir bağımsız küme bulunan kutupsal parçalanması olan, diğer adıyla tek-kutupsal, kograflar karakterize edilmiştir.

Beşinci bölüm ayrıt graflar sınıfında genelleştirilmiş boyama problemlerine ayrılmıştır. Genelleştirilmiş boyama problemlerinin, ağaçların, iki-parçalı grafların ve ayrıt-mükemmel grafların ayrıt graflarında $\mathcal{NP}$-zorluk açısından farklı davrandığı ortaya konulmuştur.

Altıncı bölümde ise, ayrıt graflar, karşılaştırmalı graflar ve genel graflar içinde genelleştirilmiş boyama problemlerinin yaklaşıklık analizi yapılmıştır. Hem standart hem de diferansiyel yaklaşım oranlarını kullanarak performans garantisi veren yaklaşım algoritmaları türetilmiştir. Yarık-boyama ve tam-boyama problemlerinin standart yaklaşım oranı açısından en az klasik boyama problemi kadar zor oldukları, ancak her ikisinin de polinomyal zamanlı diferansiyel yaklaşım şemasına, bununla birlikte klasik boyama probleminin ise sadece sabit diferansiyel yaklaşım oranına sahip oldukları gösterilmiştir. Diğer yandan, ayrık klikler eklenmesi ve tam k-parçalı bir grafın tam bağlanması işlemlerine göre kapalı olan tüm graf sınıflarında, tam-boyama probleminin yarık-boyama problemine indirgendiği gösterilmiştir.

Yedinci bölümde, yarık-boyama probleminin permütasyon graflarında iki değişik uygulaması ele alınmıştır. Bu uygulamalar yardımıyla, *eşik-boyama problemi* denen ve de diğer genelleştirilmiş boyama problemleri ile benzer şekilde incelenen yeni bir problem tanımlanmıştır.

Son bölümde, bu tez çalışmasında karşılaşılan cevabı açık olan sorulara yer verilmiştir.

Anahtar kelimeler : Genelleştirilmiş tepe boyama, Yarık-boyama, Tam-boyama, Kutupsal graf, Yaklaşım

Contents

List of Algorithms

List of Figures

List of Tables

Introduction

Combinatorial optimization problems are studied in a broad variety of fields. They are of a great use for modeling several real world problems occurring in telecommunication, transportation, distribution systems, robotics, stock management, production systems and scheduling. Indeed, setting an appropriate model for a particular problem is a difficult task; but, mostly, solving it in an efficient way appears to be even more difficult. This is why the combinatorial optimization problems are at the heart of many research topics in discrete mathematics, theoretical computer science and operations research.

Most combinatorial optimization problems of great practical relevance are shown to be extremely hard to solve; this means, with a very strong evidence, that there is no polynomial time *exact* algorithm to solve them (in an exact way). These problems are said to be $\mathcal{NP}$-*hard* (or equivalently, they belong to the complexity class of $\mathcal{NP}$-hard) and the time required to obtain their optimal solutions grows very quickly beyond the time limits of the human life. Therefore, from a practical point of view, showing that a combinatorial optimization problem is $\mathcal{NP}$-hard or not has a crucial importance.

There are several ways of handling $\mathcal{NP}$-hard problems. In the first method, one may focus on some particular cases and reduce the analysis of the problem to this framework. The question is then to know whether the problem remains $\mathcal{NP}$-hard or else, it becomes *polynomially solvable* under these specific assumptions. Detecting such restricted cases where the problem can be efficiently solved contributes to the localization of the border between "hard" and "tractable", and gives a deeper insight on the real difficulty of the problem. Secondly, one may cope with the computational hardness by aiming at obtaining some approximate solutions, that is a solution which is "close" to the optimal one, by means of some polynomial time *approximation algorithms*. The guarantee of performance of an approximate solution is then evaluated considering a worst possible case; this tells us how far the approximate solution can be from an optimal solution in the worst case. The quality of the best approximation algorithm that a problem admits, is another tool to measure the intrinsic difficulties of combinatorial optimization problems. In the third approach, one may concentrate on the sole objective of obtaining, in a reasonable time, some "good" solutions, which are not necessarily optimal but the best possible among a set of acceptable solutions for a real world application. Guiding rules of such algorithms are usually based on some intuitive ideas; thus, this field is known as *heuristics*. On the contrary to the second method, in general, no theoretical performance guarantee is known beforehand for a

heuristic; its quality is tested empirically.

In this dissertation, we adopt the first two approaches to deal with some $\mathcal{NP}$-hard problems which arise in *graph theory*. The latter is a powerful tool of extended use in operations research not only for modeling real world problems, but also for developing new methods in problem solving.

Graph theory originates with a 1736 paper "The Seven Bridges of Königsberg" by Euler [54] and since then, the field has undergone a tremendous development. Vertex coloring is one of the most extensively studied problems in graph theory for both its power of modeling a wide range of real world problems (see [36, 122]) and its theoretical interests (see for instance [10, 11, 81, 99, 23]). Given a graph, *Min (vertex) Coloring* is the problem of covering its vertex set with a minimum number of stable sets (i.e., sets of pairwise non-adjacent vertices).

In this work, our main concern is to generalize Min Coloring in order to cover an even wider field of applications. To this end, we introduce *Min Split-coloring* which is the problem of covering the vertices of a given graph with a minimum number of *split graphs*, that is, a graph whose vertex set can be partitioned into one stable set and one clique (i.e., a set of pairwise adjacent vertices). More generally, we consider the problem of covering the vertex set of a given graph with p cliques and k stable sets; several optimization problems (called *generalized coloring problems*) arising from this definition are studied, including *Min Cocoloring* [98, 53, 78]. To our knowledge, there is no study on these problems (except Min Cocoloring) in the literature; this thesis is a contribution to this field (see [50, 40, 41, 39, 42, 51]). We detect new polynomially solvable cases for Min Cocoloring and for some other problems. We also give approximation results for generalized coloring problems. In this manner, we provide a new insight on the relative difficulties of these problems and open the way to extensions of the application field of the coloring problem.

Let us briefly describe a problem occurring in robotics which can be modeled with Min Split-coloring but for which Min Coloring is not appropriate. Suppose that items to be collected are aligned along a storage corridor. There is a robot to collect these items with the constraint that the robot makes a two-way trip along the corridor and that the sizes of the items collected should be decreasing for the whole trip of the robot to ensure the stability of the pile. We assume that the robot can start its trip from either end of the corridor. The objective is to minimize the number of two-way trips that the robot makes in order to collect all the items while satisfying the constraint of decreasing sizes. One can show that this problem can be modeled as Min Split-coloring in the class of permutation graphs which is a subclass of perfect graphs. More precisely, the sizes of the items aligned on a corridor constitute a permutation. One can define the graph of this permutation by representing each item (of a certain size) by a vertex and by linking two vertices if the item having a larger size comes before the other one in the permutation. Then, the subgraph representing the items collected by the robot during one two-way trip corresponds to a split graph. This topic is studied in Chapter 7.

Let us sketch the contents of this dissertation. Chapter 1 introduces in a formal way the

generalized coloring problems and some preliminary results. Chapters 2, 3 and 5 are devoted to the detection of polynomially solvable cases in cacti and triangulated graphs, cographs, and line graphs, respectively. Chapter 4 studies several problems such as recognition and forbidden subgraph characterization, related to the polar cographs. In Chapter 6, we develop approximation algorithms for generalized coloring problems in several classes of graphs where they remain $\mathcal{NP}$-hard. Finally, Chapter 7 deals with some applications of Min Split-coloring and some problems resulting from them. We conclude in the last chapter and mention some avenues for future research.

Chapter 1

Definitions and preliminary results

1.1 Basic definitions

1.1.1 Computational Complexity

Combinatorial optimization consists of finding the best among a finite (or at least countable) set of choices. In other words, it is minimizing or maximizing a function on a discrete set while satisfying some constraints. We note that all the problems considered in this dissertation are combinatorial optimization problems.
In combinatorial optimization, the computational complexity of an algorithm is of crucial importance from a practical point of view. The theory of $\mathcal{NP}$-completeness deals with the classification of combinatorial problems in terms of the computational difficulties of the exact algorithms that they admit.
A *problem* is formally defined by a general description of its parameters and the statement of the properties that its answer, or its solution, is required to satisfy. An *instance* is just a realization of the problem parameters. We distinguish two types of problems: *decision problems* are those having an answer of the form "yes" or "no", whereas *optimization problems* require a solution of the best value, i.e., minimum or maximum value, among all possible solutions. In the latter case, a solution of the best value is called an *optimal solution.*
An *algorithm* is a step-by-step procedure giving a solution for a particular problem. An (exact) algorithm is said to *solve* a problem if it gives a true answer (or an optimal solution) for every possible instance of the problem under consideration. The *time complexity function* of an algorithm expresses the time requirements in the size of the problem, which is typically measured by the input length of the problem. It gives, for any possible size, the largest amount of time needed for the algorithm to solve a problem instance of that size.
If a problem Π admits an exact algorithm whose complexity function is polynomial, then we say that Π is "well solved" and it belongs to the class $\mathcal{P}$ of polynomially solvable problems. Roughly speaking, we say that problems for which the only known exact algorithms have an exponential time complexity (in the size of the problem) are "difficult", since the exponential growth of time requirement makes the problem intractable in a "reasonable"

amount of time. We define the class $\mathcal{NP}$ to classify these "difficult" problems; we denote by $\mathcal{NP}$ the class of decision problems that can be solved by a "non-deterministic" polynomial algorithm. In other words, $\mathcal{NP}$ is the class of decision problems for which there is a polynomial time algorithm verifying the correctness of the proof of an answer "yes". Most problems not known in $\mathcal{P}$ belong to this class. Finally, there is a class of "hardest" decision problems in $\mathcal{NP}$, denoted by *$\mathcal{NP}$-complete*; every problem in $\mathcal{NP}$ is *polynomially reduced* to every problem in $\mathcal{NP}$-complete. It means that if one could solve a problem in $\mathcal{NP}$-complete by a polynomial time algorithm A, then all the problems in $\mathcal{NP}$ could be polynomially solved by performing A only a polynomial number of times and by doing a polynomial number of elementary operations.
An optimization problem whose decision version is $\mathcal{NP}$-complete is *$\mathcal{NP}$-hard* since, obviously, it can be used to solve the associated decision problem.
The foundations for the theory of $\mathcal{NP}$-completeness were stated by S. Cook in 1971 [27]. Further developments containing a collection of $\mathcal{NP}$-complete problems can be found in [95]. The book of Garey and Johnson [70] published in 1979 remains the first reference for the theory of $\mathcal{NP}$-completeness.
Since a major part of this dissertation is devoted to the classification of several problems in $\mathcal{P}$ and $\mathcal{NP}$-complete, the reader is referred to [70] any time more information is needed on the theory of $\mathcal{NP}$-completeness, and to [28] for algorithms and time complexity computations.

1.1.2 Graph theory

In this section, we give some fairly standard definitions on graph theory. More specific notions will be defined in corresponding chapters.
A graph is nothing but a finite set of points and a set of lines joining some pairs of points. Let us put it in a more formal way as follows.

Definition 1.1 (graph). *A graph $G = (V, E)$ is defined by its vertex set $V = \{v_1, \ldots, v_n\}$ and its edge set $E = \{v_i v_j : v_i, v_j \in V\}$.*

Figure 1.1 shows an example of graph where vertices are linked by some edges.

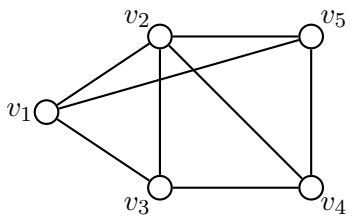


Figure 1.1: A graph with 5 vertices and 8 edges.

In this dissertation, graphs are undirected (edges are unordered pairs) and simple (no mul-

tiple edges, no loops), unless stated otherwise.
The *size* of a graph $G = (V, E)$ is the cardinality of V. We say that two vertices are *adjacent* if they are linked by an edge. A *neighbor* of a vertex v is a vertex which is adjacent to v. Then, $N(v)$ denotes the *neighborhood* of v, i.e., the set of all neighbors of v. The cardinality of $N(v)$ is called the *degree* of v and is denoted by $d(v)$. $\Delta(G)$ is the maximum degree of a vertex in G. The *non-neighborhood* of a vertex v is given by $\overline{N}(v) = V \setminus (N(v) \cup \{v\})$. Two edges are said to be *adjacent* if they have a common vertex. Also, we say that an edge v_iv_j is *incident* to v_i and v_j (or equivalently v_i and v_j are *incident* to the edge v_iv_j).
Given a graph $G = (V, E)$, the *complement* of G, denoted by $\overline{G}$, is defined by the same vertex set and two vertices in $\overline{G}$ are linked by an edge if and only if they are not linked in G.

Definition 1.2 (path). *Let $(v_1, \ldots, v_k)$ be a sequence of distinct vertices such that $v_iv_{i+1} \in E, i = 1, \ldots, k-1$. Then, it is a path of length $k-1$, denoted by P_k.*

Definition 1.3 (cycle). *Let $(v_1, \ldots, v_k)$ be a sequence of distinct vertices such that $v_iv_{i+1} \in E, i = 1, \ldots, k-1$ and $v_1v_k \in E$. Then, it is a cycle of length k, denoted by C_k.*

Definition 1.4 (clique). *Given a graph, a clique is a set of pairwise adjacent vertices.*

Definition 1.5 (stable set). *Given a graph, a stable set is a set of pairwise non-adjacent vertices.*

Given a graph G, $\omega(G)$ and $\alpha(G)$ denote respectively the size of a largest clique and the size of a largest stable set in G. $\alpha(G)$ is also called the *stability number* of G. In general, K_k and S_k denote respectively a clique and a stable set of size k; in other words a *k-clique* and respectively a *k-stable set*. Given a graph G, finding $\omega(G)$ and $\alpha(G)$, called respectively *Max Clique* and *Max Stable Set*, are $\mathcal{NP}$-hard.

Definition 1.6 (vertex coloring). *A vertex coloring (or simply coloring) of a graph $G = (V, E)$ is a function $c : V \to \mathcal{C}$ from the set of vertices to a set of colors (which is typically represented by a set of integer numbers) in such a way that $c(v_i) \neq c(v_j)$ whenever $v_iv_j \in E$. A k-coloring of G is a coloring using at most k colors.*

If a graph G admits a k-coloring, then we say that G is *k-colorable*.

Definition 1.7 (Min Coloring). *Given a graph G, Min Coloring is the problem of finding the smallest k such that G is k-colorable.*

The optimal value of Min Coloring is called *chromatic number* and is denoted by $\chi(G)$. Note that, in a graph coloring, each color class corresponds to a stable set. Hence, minimum coloring of a graph $G = (V, E)$ can be seen as a partition of V into a minimum number of stable sets. 3-colorability is known to be $\mathcal{NP}$-complete [71] implying that Min Coloring is $\mathcal{NP}$-hard in general.

Definition 1.8 (Min Clique Cover). *Given a graph $G = (V, E)$, Min Clique Cover is the problem of finding the smallest number of cliques partitioning V.*

The optimal value of Min Clique Cover, which is also $\mathcal{NP}$-hard, is called *clique cover number* and is denoted by $\theta(G)$.
For a set $V' \subset V$, $G[V']$ denotes the subgraph of G *induced* by V', meaning that $G[V'] = (V', E')$ where $E' = \{v_i v_j : v_i \in V', v_j \in V' \text{ and } v_i v_j \in E\}$. Whereas, a *partial* subgraph of G has a vertex set and an edge set, which are respectively subsets of the vertex set and the edge set of G. In what follows, a subgraph H of G, denoted by $H \subseteq G$, is always an induced subgraph of G, unless stated otherwise. Note that we sometimes denote $G[V \setminus \{v\}]$ and $G[V \setminus V']$ respectively by $G \setminus v$ and $G \setminus V'$, for the sake of simplicity.
Let G and H be two graphs. We say that G is *H-free* if G does not contain H as an induced subgraph.
In the analysis of problems such as Min Clique Cover, Min Coloring, Max Clique and Max Stable Set, there is a particularly important class of graphs, called perfect graphs.

Definition 1.9 (Perfect graphs). *A graph G is perfect if and only if for all $H \subseteq G$, we have $\chi(H) = \omega(H)$.*

In particular, for the class of perfect graphs, the stability number as well as the chromatic number can be determined in polynomial time (see [81]). It was shown by Lovász in 1972 [99] that a graph G is perfect if and only if its complement is perfect (this was known as the weak perfect graph conjecture (see [11]). Therefore, a graph G is perfect if and only if for all $H \subseteq G$, we have $\theta(H) = \alpha(H)$. Let us also state the following theorem known as the Strong Perfect Graph Conjecture from 1961 (formulated by Berge [10]) until it has been proven by Chudnowsky, Robertson, Seymour and Thomas in 2002 [23].

Theorem 1.10 ([23]). *A graph G is perfect if and only if it does not contain any odd hole (i.e., $C_{2k+1}, k \geq 2$), or odd antihole (i.e., the complement of $C_{2k+1}, k \geq 2$) as an induced subgraph.* □

Note that in this dissertation, we will be dealing mostly, but not always, with subclasses of perfect graphs.
Given a graph, the problem of covering its vertex (or edge) set by disjoint sets (with some property) is called a *partitioning problem*, whereas finding a vertex (or edge) set of maximum size verifying some property is a *packing problem*.
Each time we mention that a set is *maximal* or *minimal*, we mean inclusionwise; a set L is maximal (respectively minimal) with respect to a certain property P if P holds for L but the addition (respectively the removal) of an element transforms L into a set not verifying P. Whereas the terms *maximum* and *minimum* refer to the size of a set.
A graph G is called *connected* if there is a path linking any pair of vertices in G. A *connected component* of G is then a maximal connected subgraph of G. A graph which is not connected is called *disconnected*.

Let H be a graph, we denote by kH a graph consisting of k connected components isomorphic to H. In other words, kH is the graph obtained by taking *disjoint union* of k copies of H; it is denoted by $kH = \cup_{i=1}^{k} H_i$, where each H_i is isomorphic to H. We also define the *join* of two graphs G_1 and G_2; it is the graph $\overline{\overline{G_1} \cup \overline{G_2}}$ denoted by $G_1 \oplus G_2$.
A class of graphs $\mathcal{G}$ is called *hereditary* if every subgraph of a graph in $\mathcal{G}$ also belongs to $\mathcal{G}$. Note that in this dissertation, only hereditary classes of graphs are considered, unless otherwise stated.
In what follows, for a given graph $G = (V, E)$, we have $|V| = n$ and $|E| = m$. See [11] for all graph theoretical notions not given here.

1.2 Generalized vertex coloring problems

1.2.1 Definitions and basic properties

The problems we deal with in this dissertation are all some generalizations of Min Coloring. Roughly speaking, given a graph, we want to cover its vertex set not only with stable sets but also with cliques. We obtain different problems according to the objective function we choose. In this section, we define the main problems commonly studied in all chapters; namely (p, k)-coloring problems, Min Split-coloring and Min Cocoloring. We also introduce the notion of polar graphs, exclusively studied in Chapter 4, which is a particular way of partitioning the vertices of a given graph into cliques and stable sets. In the sequel, let us present our basic problem in the most general form.

Definition 1.11 ((p,k)-coloring). *A graph $G = (V, E)$ is (p, k)-colorable if V can be partitioned into p cliques and k stable sets of G, i.e., $V = \left(\cup_{i=1}^{p} K^i \right) \bigcup \left(\cup_{j=1}^{k} S^j \right)$ where K^i's are cliques and S^j's are stable sets. Such a partition $(K^1, \ldots, K^p, S^1, \ldots, S^k)$ is called a (p, k)-coloring of G.*

We denote by $\mathcal{K}_p\mathcal{S}_k$ the class of (p, k)-colorable graphs. Obviously, if $G \in \mathcal{K}_p\mathcal{S}_k$ then $G \in \mathcal{K}_{p'}\mathcal{S}_{k'}$ for all $p' \geq p$ and $k' \geq k$. Given a class $\mathcal{G}$ of graphs, $\mathcal{G} \cap \mathcal{K}_p\mathcal{S}_k$ is said to be *polynomially determined* if there exists a polynomial time algorithm computing, for every graph $G \in \mathcal{G}$, a (p, k)-coloring of G whenever $G \in \mathcal{K}_p\mathcal{S}_k$ or recognizing that $G \notin \mathcal{K}_p\mathcal{S}_k$. Here are the two optimization problems that we call *(p, k)-coloring problems.*

Definition 1.12 ((p_{min},k)-coloring). *Given a graph G and an integer k, (p_{min}, k)-coloring is the minimization problem of finding*

$$p_{min}(G) = \min\{p : G \in \mathcal{K}_p\mathcal{S}_k\}.$$

In a similar way we define the symmetric problem.

Definition 1.13 ((p,k_{min})-coloring). *Given a graph G and an integer p, (p, k_{min})-coloring is the minimization problem of finding*

$$k_{min}(G) = \min\{k : G \in \mathcal{K}_p\mathcal{S}_k\}.$$

The recognition of (p,k)-colorable graphs has been already studied in the literature [16, 18, 58, 96]. Nevertheless, the first results on the corresponding optimization problems that we call (p,k)-coloring problems are given in [40, 41] and gathered in this thesis.
Let us now define the problem of Min Split-coloring which is our main concern in this dissertation.

Definition 1.14 (Min Split-coloring). *Given a graph G, Min Split-coloring consists in finding*

$$\chi_S(G) = \min\{\max(p,k) : G \in \mathcal{K}_p\mathcal{S}_k\} = \min\{k : G \in \mathcal{K}_k\mathcal{S}_k\}$$

where $\chi_S(G)$ is called the split-chromatic number of G. A partition of the vertex set of G into k cliques and k stable sets is called a k-split-coloring of G.

Note that the second term defining $\chi_S(G)$ is equivalent to the first one since in a split-coloring, one may always suppose that there are empty cliques or stable sets.

Definition 1.15 (Split graph). *A graph $G = (V,E)$ is a split graph if its vertex set V can be partitioned into a clique K and a stable set S. The pair (K,S) is called a split partition.*

The reason why the problem defined in Definition 1.14 is called Min Split-coloring is that, it amounts to covering the vertex set of a given graph by a minimum number of split graphs. Min Split-coloring was recently defined in [50] by Ekim and de Werra. Later on, various papers giving first results on Min Split-coloring were published [40, 41, 44]; they are all integrated in this dissertation.
Finally, the last problem we deal with consists in minimizing the total number of cliques and/or stable sets covering all the vertices of a graph.

Definition 1.16 (Min Cocoloring). *Given a graph G, Min Cocoloring is the problem of finding*

$$z(G) = \min\{p + k : G \in \mathcal{K}_p\mathcal{S}_k\}$$

where $z(G)$ is called the cochromatic number of G. A partition of the vertex set of G into p cliques and k stable sets is called a $(p+k)$-cocoloring of G.

Contrary to Min Split-coloring, Min Cocoloring is widely studied in the literature; first defined in [98] by Lesniak and Straight, $\mathcal{NP}$-completeness proofs and polynomial cases are given in [76, 78, 117], approximation results are derived in [65] and extremal graph theoretical results are given in [53, 77].
In all these generalizations of Min Coloring, taking cliques and stable sets together has the nice property of complementarity: the problem is basically the same whether we consider the graph G or its complement $\overline{G}$. In fact, $G \in \mathcal{K}_p\mathcal{S}_k$ if and only if $\overline{G} \in \mathcal{K}_k\mathcal{S}_p$. This implies that $z(G) = z(\overline{G}), \chi_S(G) = \chi_S(\overline{G}), p_{min}(G) = k_{min}(\overline{G})$ and $k_{min}(G) = p_{min}(\overline{G})$ (for $k = p$). Restricted to Min Split-coloring, we come up with the *auto-complementarity* property, which means that solving Min Split-coloring in a graph is equivalent to solve it in its complement. Moreover, (p,k)-coloring problems, Min Split-coloring and Min Cocoloring are all *hereditary*

since any induced subgraph of a split graph, a clique and a stable set are respectively a split graph, a clique and a stable set. It follows that if a graph G is (p,k)-colorable then for all $H \subseteq G, H$ is also (p,k)-colorable.
We also associate a packing problem with the (p,k)-coloring of a graph.

Definition 1.17 (Max (p,k)-colorable subgraph). *Given a graph G, Max (p,k)-colorable subgraph consists in finding an induced (p,k)-colorable subgraph of maximum size in G. Its optimal value is denoted by $\alpha_{p,k}(G)$.*

By convention, $\alpha_{0,0}(G) = 0$. Also, by definition, $\alpha_{1,0}(G) = \omega(G)$ and $\alpha_{0,1}(G) = \alpha(G)$. Note that $\alpha_{1,1}(G)$, called the *split-independence number* of G and also denoted by $\alpha_S(G)$, is the size of a largest induced split graph in G.
We remark that in the context of (p,k)-coloring problems and Min Cocoloring, a color corresponds to either a clique or a stable set whereas a color is equivalent to a split graph in the case of split-coloring.
One may immediately derive some natural relations between the optimal values of the above problems: we have clearly $\chi_S(G) \leq z(G)$ and also $z(G) \leq \min(\chi(G), \theta(G))$ since any coloring or covering by cliques is also a cocoloring. Furthermore, we have $z(G) \leq 2\chi_S(G)$ because a split-coloring with $\chi_S(G)$ colors is a cocoloring with $2\chi_S(G)$ colors. Finally, the relation $\alpha_{p,0}(G) + \alpha_{0,k}(G) - pk \leq \alpha_{p,k}(G) \leq \alpha_{p,0}(G) + \alpha_{0,k}(G)$ holds since p cliques and k stable sets may meet in at most pk vertices.
Now, let us introduce polar graphs which are a natural extension of some classes of graphs like bipartite graphs, split graphs and complements of bipartite graphs. Their definition follows from [21].

Definition 1.18 (Polar graph). *A graph $G = (V, E)$ is called polar if its vertex set V can be partitioned into (A, B) (A or B may possibly be empty) such that A induces a complete multipartite graph (it is a join of stable sets) and B a (disjoint) union of cliques (it is the complement of a join of stable sets).*

It can be easily seen that polar graphs admit a particular (p,k)-coloring for some p and k; complete links among stable sets and no links among cliques are two conditions that such a (p,k)-coloring should verify. We also use the following definition.

Definition 1.19 ((s,t)-polar). *A graph is (s,t)-polar if it admits a polar partition (A, B), where A is a join of at most s stable sets and B is a union of at most t cliques.*

It can be easily noticed that $(1,1)$-polar graphs are just split graphs.
In this section, we will restrain ourselves to this introduction on polar graphs. Since polar graphs are exclusively studied in Chapter 4, their definition will be recalled and their basic properties will be discussed in a brief introduction of that chapter.

1.2.2 General framework of our study

Obviously, the decision versions of all the generalized coloring problems mentioned above are $\mathcal{NP}$-complete in arbitrary graphs. Considering 3-colorability which is $\mathcal{NP}$-complete, it is easily seen that deciding whether a graph is (p,k)-colorable is $\mathcal{NP}$-complete for $p \geq 3$ or $k \geq 3$. It follows immediately that finding $\alpha_{p,k}$ is also $\mathcal{NP}$-hard. Moreover, Min Cocoloring and Min Split-coloring are $\mathcal{NP}$-hard since they contain as a special case both Min Coloring and Min Clique Cover. Later in this chapter, more precisely in Corollary 1.39, we will also see that there is a class of graphs, where, although $\alpha_{1,0}$, $\alpha_{0,1}$, $\alpha_{0,2}$ and $\alpha_{0,3}$ can be computed in polynomial time, it is $\mathcal{NP}$-hard to find $\alpha_{1,1}$. In what follows, since all the generalized coloring problems are clearly in $\mathcal{NP}$, we simply omit this in some of the $\mathcal{NP}$-completeness proofs.

Several extensions of the usual vertex coloring problem have been introduced during the last decades. One of the most relevant to our study is the problem of coloring the vertices of a given graph in such a way that each color class verifies a given property P. This problem originated with Folkman [64] and was extensively studied since then (see [20] for the state of the art on this topic). In [20], Brown considered the complexity status of this problem for various properties; he paid a special attention to the property of not containing an induced subgraph isomorphic to H, where H is a graph fixed from the beginning. This is referred to as *H-free coloring* in the literature. Along the same line, Achlioptas showed that H-free k-coloring is $\mathcal{NP}$-complete for any graph H other than K_2 and $\overline{K_2}$ for all $k \geq 2$ [2].

A generalization of H-free coloring is due to Kratochvíl et al. [97]; a graph is $(\mathcal{P}_1, \ldots, \mathcal{P}_p)$-*colorable* if it admits a partition $(U_1, \ldots, U_p)$ of its vertex set, where each U_i verifies the property $\mathcal{P}_i$. Kratochvíl et al. conjectured in [97] that for any additive hereditary graph properties $\mathcal{P}_i$, recognizing $(\mathcal{P}_1, \ldots, \mathcal{P}_p)$-colorable graphs is $\mathcal{NP}$-complete with the obvious exception of bipartite graphs. Note that a property P is *additive* if for any two vertex-disjoint graphs verifying P, their union also verifies P. They also proved the conjecture in some special cases. Several subcases have already been settled by various authors (see [55]). Farrugia proved this conjecture in [55] for all induced-hereditary additive properties. More generally, a graph is called $(\mathcal{P}_1, \ldots, \mathcal{P}_p, \mathcal{Q}_1, \ldots, \mathcal{Q}_k)$-*colorable* if its vertex set can be partitioned into sets $(U_1, \ldots, U_p, W_1, \ldots, W_k)$ where each U_i verifies the co-additive property $\mathcal{P}_i$ and each W_j verifies the additive property $\mathcal{Q}_j$ (a property P is *co-additive* if its complement, that is the property verified by the complement of the graphs with property P, is additive). Recently, Alekseev et al. discussed in [3] the complexity status of recognizing $(\mathcal{P}_1, \ldots, \mathcal{P}_p, \mathcal{Q}_1, \ldots, \mathcal{Q}_k)$-colorable graphs where $\mathcal{P}_i$'s and $\mathcal{Q}_j$'s are also hereditary properties. In this case, the authors show that recognizing $(\mathcal{P}_1, \ldots, \mathcal{P}_p, \mathcal{Q}_1, \ldots, \mathcal{Q}_k)$-colorable graphs is $\mathcal{NP}$-complete whenever the recognition of $(\mathcal{P}_1, \ldots, \mathcal{P}_p)$-colorable graphs or the recognition of $(\mathcal{Q}_1, \ldots, \mathcal{Q}_k)$-colorable graphs is $\mathcal{NP}$-complete. The $\mathcal{NP}$-completeness of (p,k)-coloring problems for $p \geq 3$ or $k \geq 3$ also follows from this result; it suffices to consider $\mathcal{P}_i$'s as $\overline{K_2}$-free, and $\mathcal{Q}_j$'s as K_2-free.

As it will be explained at the end of this subsection, our methodology differs from these studies where the focus is on the properties a color class is required to verify and the com-

plexity status is settled for various types of properties. We will rather fix the problems on which we work and decide their complexities in some restricted cases. Before explaining the framework of our study, let us point out some facts on the relative difficulties of the generalized coloring problems that we have introduced.

Remark 1.20. *Min Split-coloring and Min Cocoloring are reduced to (p,k)-coloring problems in the class of all finite graphs.*

Remark 1.21. *(p_{min},k)-coloring is reduced to (p,k_{min})-coloring and vice versa in the class of all finite graphs.*

Notice that a polynomial time algorithm for (p_{min},k)-coloring or (p,k_{min})-coloring (for any given p and k) would immediately imply polynomial time algorithms for Min Split-coloring and Min Cocoloring. In fact, solving one of the general (p,k)-coloring problems, say for instance (p,k_{min})-coloring (respectively (p_{min},k)-coloring), for at most n (but in reality much less) different values of p, one can solve (p_{min},k)-coloring (respectively (p,k_{min})-coloring), Min Split-coloring and Min Cocoloring by choosing the value of p which allows to optimize the objective function of the problem under consideration. In other words, (p,k)-coloring problems are more difficult than Min Split-coloring and Min Cocoloring and moreover, (p,k_{min})-coloring and (p_{min},k)-coloring always belong to the same complexity class independently from the class of graphs in which we restrict our analysis. In return, there is no such comparison between Min Split-coloring and Min Cocoloring; this topic will be discussed in more details in Chapters 5, 6 and 7.

Although split graphs have been extensively studied by many authors (see [62, 63, 82, 9, 22]), Min Split-coloring does not seem to have been studied to our knowledge. The only coloring problem related to split graphs which appears extensively in the literature deals with the case where the edge set of a graph has to be covered [22, 100]. However, this problem of edge coloring by split graphs has neither the hereditary character nor the auto-complementarity property mentioned above.

In a similar way, to our knowledge, the optimization problems of (p,k)-coloring, namely (p_{min},k)-coloring and (p,k_{min})-coloring, were not defined before [40] although they appear implicitely in some algorithms of [78] and [87] .

This dissertation consists in a systematic study of generalized coloring problems defined in this section by restricting our analysis to different classes of graphs. Each time, either we derive polynomial time algorithms solving efficiently the problem under consideration, or we show that, even under these particular assumptions, the problem we handle remains $\mathcal{NP}$-hard. In the latter case, an approximation algorithm will be derived and its approximation ratio will be analyzed. When appropriate, non approximability results will also be established. Then, our purpose will be to compare the behavior of each problem in order to determine in a best possible way the limits between $\mathcal{NP}$-completeness and polynomiality, and also to study their relative computational difficulties (in terms of admitting polynomial time exact algorithms or the quality of the approximation ratios).

Getting back to polar graphs, it was shown in [21] that recognizing whether an arbitrary

graph is polar is $\mathcal{NP}$-complete. In our work, we detect the first class of graphs, to our knowledge, where the recognition of polar graphs becomes polynomial.

1.3 2-split-colorability

The $\mathcal{NP}$-completeness of k-split-coloring for $k \geq 3$ was stated in the previous section. In the sequel, we handle the problem of 2-split-colorability and we show that it is polynomially solvable in all finite graphs. Before adopting an algorithmic approach, we first attempt to give a forbidden subgraph characterization of 2-split-colorable graphs; we derive partial results in this direction.
Now, let us introduce the notion of split-critical graphs.

Definition 1.22 (k-split-critical). *A graph $G = (V, E)$ is called k-split-critical if $\chi_S(G) = k$ and for all $v \in V$, $\chi_S(G[V \setminus v]) = k - 1$.*

We say that a graph is *triangulated*, or equivalently *chordal*, if every cycle of length at least 4 contains a *chord*, i.e., an edge linking two non-adjacent vertices of the cycle. Let us state the following theorem characterizing split graphs.

Theorem 1.23 ([62]). *For every graph $G = (V, E)$, the following conditions are equivalent:*

1. *G is a split graph;*
2. *G and $\overline{G}$ are triangulated;*
3. *G does not contain $2K_2$, C_4 or C_5.* □

According to this definition, $2K_2$, C_4 and C_5 are obviously the unique 2-split-critical structures. Moreover, the recognition of split graphs can be done by a $\mathcal{O}(n^2)$ algorithm given in [100]. Also, a recent result in [87] shows that split graphs can be recognized by a linear algorithm in time $\mathcal{O}(n + m)$; this topic will be developed in Section 2.2.
In order to characterize some 3-split-critical graphs and as a preliminary for further developments, we mention the following three facts. In what follows, we denote by OC an odd cycle, i.e., a cycle of length $2k + 1$ with $k \geq 1$.

Fact 1.24. *For any odd cycle OC of length at least 5, we have $\chi_S(OC) = 2$; a 2-split-coloring is obtained by choosing one clique (of size one or two) and two stable sets.*

Fact 1.25. *For any two induced odd cycles $2OC$ of length at least 5, we have $\chi_S(2OC) = 2$; a 2-split-coloring is obtained by choosing one clique on each cycle and two stable sets in the remaining graph.*

Fact 1.26. *The graph $3OC$ consisting of three induced odd cycles is not 2-split-colorable, i.e., $\chi_S(3OC) \geq 3$.*

Fact 1.26 follows from the fact that each clique can contain vertices of at most one cycle and that an odd cycle is not 2-colorable in the classical sense. Therefore, we have to assign a new color to at least one vertex of the third odd cycle as shown in Figure 1.2. Note that in all figures representing a split-coloring, cliques are encircled so that one may distinguish them from the stable sets of same color.

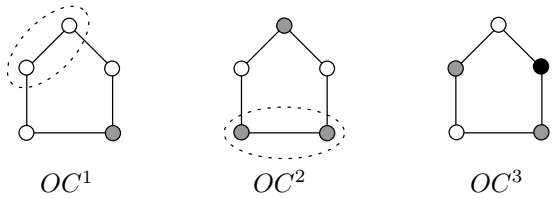


Figure 1.2: $3OC$ is not 2-split-colorable.

Proposition 1.27. *A graph G consisting of three induced odd cycles $3OC$ or its complement $\overline{G} = \overline{3OC}$ is 3-split-critical.*

Proof. It suffices to notice that the way one can partition the vertices of a $3OC$ as in Figure 1.2 is the best possible and that in this way, the inequality of Fact 1.26 turns to be an equality. The criticality of $3OC$ is just routine to check. In addition, the auto-complementarity of the split-coloring gives the result. □

The above proposition implies in particular that $3K_3$ and $\overline{3K_3}$ are 3-split-critical graphs. Now, let us open a parenthesis on k-split-critical graphs for $k \geq 4$ and generalize this result.

Proposition 1.28. *kK_k, disjoint union of k cliques of size k, and $\overline{kK_k}$, join of k stable sets of size k with complete links, are k-split-critical.*

Proof. One sees immediately that kK_k is not $(k-1)$-split-colorable. We obtain a k-split-coloring of kK_k by assigning repetitively a new color to a split graph consisting of an inclusion-wise maximal uncolored (sub-)clique of a K_k and a stable set having one vertex in each remaining maximal (sub-)clique. One may observe that removing any vertex makes our graph $(k-1)$-split-colorable. In addition, the criticality of $\overline{kK_k}$ is obtained by the auto-complementarity of the split-coloring. □

Note that mK_k with arbitrary large m is still k-split-colorable, i.e., we do not increase χ_S by adding new disjoint K_k's, but it is no more k-split-critical. This fact can be observed for mK_4, with $m > 4$ in Figure 1.3.

Let us construct another k-split-critical graph which is also triangle-free. It makes use of *Mycielski graphs* that are constructed iteratively; we denote by M_k the Mycielski graph of iteration k. We start with M_2 which consists of two vertices linked by an edge. At iteration k, we take one copy v' of each vertex v in M_{k-1} and we link v' to all neighbors of v. Then we add one more vertex z_k linked to all vertices introduced at iteration k. One can show

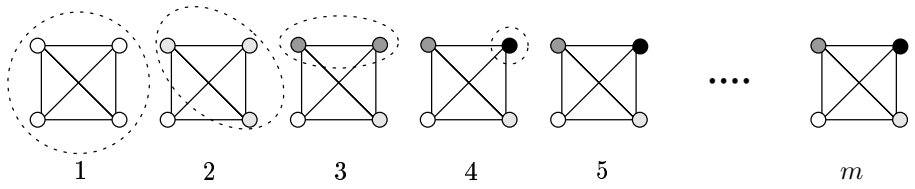


Figure 1.3: mK_4 is 4-split-colorable.

that for all k, M_k is triangle-free and $\chi(M_k) = k$, moreover, it is k-critical in the usual sense (i.e., the removal of any vertex yields a $(k-1)$-colorable graph) [105].

Proposition 1.29. *kM_k and $\overline{kM_k}$, where M_k is the Mycielski graph of iteration k, are k-split-critical.*

Proof. A k-split-coloring of kM_k is obtained by taking at most k cliques (one from each disjoint M_k) and by partitioning the remaining vertices into k (or $k-1$, if exactly k cliques are set) stable sets. Moreover, for all $v \in kM_k$, we have $\chi_S(kM_k \setminus v) = k-1$. An optimal split-coloring is obtained by taking $k-1$ cliques, each one consisting in z_k in a M_k except the one where the vertex v is removed, and $k-1$ stable sets covering $M_k \setminus v$ and $(k-1)(M_k \setminus z_k)$ (which is possible since M_k is k-critical). Moreover, kM_k is not $(k-1)$-split-colorable since each M_k is not $(k-1)$-colorable unless we remove a clique containing z_k and there are exactly k of them. The k-split-criticality of $\overline{kM_k}$ follows from the autocomplementarity of k-split-coloring. □

Let us go back to the case of 2-split-colorability. Proposition 1.27 means that whenever a graph G contains three induced odd cycles or its complement, the split-chromatic number of G is at least 3. However, the number of induced odd cycles do not play any key role in the determination of χ_S for $\chi_S \geq 3$ because $\chi_S(mOC) = 3$, $\forall m \geq 3$.
We know that the 3-split-critical graphs of Proposition 1.27 do not form a complete list of forbidden subgraphs characterizing 2-split-colorable graphs. For instance, a particular C_5-cactus described in Chapter 2, Section 2.1.2 (see Figure 2.4 a)) is another 3-split-critical graph not containing $3OC$. Several other examples can be easily stated, for instance, the graph in Figure 1.4 on which a possible 3-split-coloring is represented. It would be interesting to be able to complete this (not finite) list of forbidden subgraphs to characterize 2-split-colorable graphs.

One can observe that the 2-split-colorability problem is closely related to the graph bipartization problem (see [61] for a survey), where we are looking for a minimum vertex set such that its removal gives a bipartite graph. In return, in our problem, we relax the minimality requirement and instead, we impose on the vertex set hitting every odd cycle to have the property of being covered by two cliques.
Let us describe the basic idea of a polynomial time algorithm for any graph, which ends, if any, with a $(2,2)$-coloring of a given graph and, if not, with a negative answer. To this

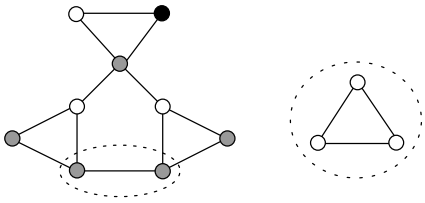


Figure 1.4: A 3-split-critical graph.

purpose, we need to be able to decide whether a given subgraph is in $\mathcal{K}_1\mathcal{S}_2$ or in $\mathcal{K}_2\mathcal{S}_1$. In the first part of the algorithm recognizing a $(1,2)$-colorable graph, we set a necessary partition of vertices by applying a test on the neighborhood and on the non-neighborhood of every vertex. In fact, in a $(1,2)$-coloring of a graph, if a vertex x is in a clique then $\overline{N}(x) \in \mathcal{K}_0\mathcal{S}_2$, and if x is in a stable set then $N(x) \in \mathcal{K}_1\mathcal{S}_1$. It implies that for a vertex x, if $\overline{N}(x) \notin \mathcal{K}_0\mathcal{S}_2$ then necessarily x is in a stable set and, similarly, if $N(x) \notin \mathcal{K}_1\mathcal{S}_1$ then necessarily x is in a clique. The second part consists in finding a final $(2,2)$-coloring via testing only a polynomial number of modifications on the necessary partition. Note that the detection of $(1,2)$-colorable and $(2,1)$-colorable graphs are given in [16] and detailed in [19]. In return, the complete proof for 2-split-colorability with its exact complexity has never been published; Brandstädt has just mentioned in [19] that this problem is also polynomial and that details can be found in his technical reports [15]. This method enables us to decide the 2-split-colorability of any graph in polynomial time[1]. We do not detail this algorithm here; instead, we propose, in the next section, the generalization of this approach in order to obtain an algorithm deciding (under some conditions) the (p,k)-colorability of general graphs for any fixed p and k.

1.4 (p,k)-coloring of general graphs

In this section, we emphasize some particular cases where (p,k)-colorability can be polynomially determined: we will observe that this happens for fixed p and k together with some additional conditions.

We have already seen that (p,k)-colorable graphs for $(1,2)$ and $(2,1)$ are shown to be polynomially determined [19]. Besides, the cases for $(1,0)$, $(0,1)$, $(1,1)$, $(2,0)$ and $(0,2)$ are obviously known to be polynomially determined. Let us recall that a graph G is in $\mathcal{K}_p\mathcal{S}_k$ if and only if $\overline{G}$ is in $\mathcal{K}_k\mathcal{S}_p$; consequently, for every class of graphs $\mathcal{C}$ closed under complementation, $\mathcal{K}_p\mathcal{S}_k \cap \mathcal{C}$ is polynomially determined if and only if $\mathcal{K}_k\mathcal{S}_p \cap \mathcal{C}$ is polynomially determined.

The main idea of the algorithm of (p,k)-coloring of general graphs is based on the following remark.

[1] The exact time complexity is polynomial with a high degree.

Remark 1.30. *If $(K^1, \ldots, K^p, S^1, \ldots, S^k)$ is a (p,k)-coloring of G, where $p, k \geq 1$, then for all $x \in \cup_{j=1}^{k} S^j$, $G[N(x)]$ is $(p, k-1)$-colorable, and for all $x \in \cup_{i=1}^{p} K^i$, $G[\overline{N}(x)]$ is $(p-1, k)$-colorable.*

Here we give an explicit form of an algorithm deciding whether a given graph is in $\mathcal{K}_p\mathcal{S}_k$ or not and, whenever it is, giving a (p,k)-coloring of the graph. Then, polynomially solvable cases will be exhibited from this exact algorithm.

Algorithm 1 (p,k)-coloring of general graphs

Input: graph G, $(p,k) \in \mathbb{N} \times \mathbb{N}$

Output: a (p,k)-coloring of G if any, answer "G is not in $\mathcal{K}_p\mathcal{S}_k$" otherwise

1. $\mathcal{S}' \longleftarrow \emptyset$, $\mathcal{K}' \longleftarrow \emptyset$;
2. **Phase 1**
3. **for all** $x \in V$ **do**
4. **if** $G[\{x\} \cup \overline{N}(x)] \notin \mathcal{K}_{p-1}\mathcal{S}_k$ **then**
5. $\mathcal{S}' \longleftarrow \mathcal{S}' \cup \{x\}$;
6. **else**
7. **if** $G[N(x)] \notin \mathcal{K}_p\mathcal{S}_{k-1}$ **then**
8. $\mathcal{K}' \longleftarrow \mathcal{K}' \cup \{x\}$;
9. **if** $G[\mathcal{S}'] \notin \mathcal{K}_0\mathcal{S}_k$ or $G[\mathcal{K}'] \notin \mathcal{K}_p\mathcal{S}_0$ **then**
10. **print** "G is not in $\mathcal{K}_p\mathcal{S}_k$", **exit**.
11. **Phase 2**
12. **if** $G[V \setminus \mathcal{S}'] \in \mathcal{K}_p\mathcal{S}_0$ **then**
13. deduce a (p,k)-coloring, **exit**.
14. **else**
15. **for all** $x \notin \mathcal{K}' \cup \mathcal{S}'$ **do**
16. $[(K_x^1, \ldots, K_x^p, S_x^1, \ldots, S_x^{k-1})] \longleftarrow$ a $(p, k-1)$-coloring of $G[N(x)]$;
17. $[(K_{\bar{x}}^1, \ldots, K_{\bar{x}}^{p-1}, S_{\bar{x}}^1, \ldots, S_{\bar{x}}^k)] \longleftarrow$ a $(p-1, k)$-coloring of $G[\{x\} \cup \overline{N}(x)]$;
18. **for all** $(W_x, Z_x, W_{\bar{x}}, Z_{\bar{x}})$ such that
$W_x \subseteq \cup_{j=1}^{k-1} S_x^j,\ |W_x| \leq (k-1)p$
$Z_x \subseteq N(x) \setminus \cup_{j=1}^{k-1} S_x^j,\ |Z_x| \leq (k-1)p$
$W_{\bar{x}} \subseteq \cup_{j=1}^{k} S_{\bar{x}}^j,\ |W_{\bar{x}}| \leq pk$
$Z_{\bar{x}} \subseteq \{x\} \cup \overline{N}(x) \setminus \cup_{j=1}^{k} S_{\bar{x}}^j,\ |Z_{\bar{x}}| \leq (p-1)k$ **do**
19. $\mathcal{S}'' \longleftarrow \mathcal{S}' \cup (\cup_{j=1}^{k-1} S_x^j \setminus W_x) \cup Z_x \cup (\cup_{j=1}^{k} S_{\bar{x}}^j \setminus W_{\bar{x}}) \cup Z_{\bar{x}}$;
20. **if** $G[\mathcal{S}''] \in \mathcal{K}_0\mathcal{S}_k$ and $G[V \setminus \mathcal{S}''] \in \mathcal{K}_p\mathcal{S}_0$ **then**
21. deduce a (p,k)-coloring of G, **exit**.
22. **print** "G is not in $\mathcal{K}_p\mathcal{S}_k$", **exit**.

Theorem 1.31. *Let $\mathcal{G}$ be a hereditary class of graphs, let $(p,k) \in \mathbb{N} \times \mathbb{N}$, $p, k \geq 1$. If $\mathcal{K}_p\mathcal{S}_{k-1} \cap \mathcal{G}$, $\mathcal{K}_{p-1}\mathcal{S}_k \cap \mathcal{G}$, $\mathcal{K}_0\mathcal{S}_k \cap \mathcal{G}$ and $\mathcal{K}_p\mathcal{S}_0 \cap \mathcal{G}$ are polynomially determined, then $\mathcal{K}_p\mathcal{S}_k \cap \mathcal{G}$ is also polynomially determined by Algorithm 1.*

Proof. Algorithm 1 gives a (p,k)-coloring of a given graph in polynomial time when the hypotheses of the theorem are satisfied. Remark 1.30 implies that **Phase 1** works by necessary conditions. This means that if G is (p,k)-colorable then vertices placed in $\mathcal{K}'$ (respectively $\mathcal{S}'$) during the first phase necessarily belong to some clique (respectively stable set) in every (p,k)-coloring of G. Consequently $G \in \mathcal{K}_p\mathcal{S}_k$ if and only if $\mathcal{K}'$ and $\mathcal{S}'$ can be completed to a partition $(\mathcal{K},\mathcal{S})$ of V, where $G[\mathcal{K}] \in \mathcal{K}_p\mathcal{S}_0$ and $G[\mathcal{S}] \in \mathcal{K}_0\mathcal{S}_k$. This is namely the aim of **Phase 2** where the existence of such a completion is tested through any possible partition of vertices respecting $\mathcal{S}'$ and $\mathcal{K}'$. Note that every vertex $x \in V \setminus (\mathcal{K} \cup \mathcal{S})$ satisfies $G[\{x\} \cup \overline{N}(x)] \in \mathcal{K}_{p-1}\mathcal{S}_k$ and $G[N(x)] \in \mathcal{K}_p\mathcal{S}_{k-1}$. A (p,k)-coloring of G is detected in line 13 if the set of vertices not included in $\mathcal{S}'$ can be partitioned into p cliques, given the fact that $\mathcal{S}' \in \mathcal{K}_0\mathcal{S}_k$.
If line 13 is not executed, then, in every (p,k)-coloring of G (if any), at least one vertex of $V \setminus (\mathcal{K}' \cup \mathcal{S}')$, say x_0, belongs to a stable set. Therefore, $(\mathcal{K} \cup \mathcal{S}) \cap N(x_0) \in \mathcal{K}_p\mathcal{S}_{k-1}$. Then, let $(K_1^{x_0}, \ldots, K_p^{x_0}, S_1^{x_0}, \ldots, S_{k-1}^{x_0})$ and $(K_1^{\bar{x}_0}, \ldots, K_{p-1}^{\bar{x}_0}, S_1^{\bar{x}_0}, \ldots, S_k^{\bar{x}_0})$ respectively be a $(p,k-1)$-coloring of $G[N(x_0)]$ and a $(p-1,k)$-coloring of $G[\{x_0\} \cup \overline{N}(x_0)]$.
Since every stable set intersects $\mathcal{K}$ in at most p vertices, and every clique intersects $\mathcal{S}$ in at most k vertices, we obtain $\mathcal{S}'' = \mathcal{S}$ for an execution of line 19. Then, a (p,k)-coloring of G is detected during the related execution of line 21, which concludes the proof. □

We remark that Algorithm 1 gives a priority to stable sets. Note that line 13 can be replaced by a test on $\mathcal{K}'$ and not on $\mathcal{S}'$; namely, if $G[V \setminus \mathcal{K}'] \in \mathcal{K}_0\mathcal{S}_k$ then deduce a (p,k)-coloring of G. In such a "clique version", $(W_x, Z_x, W_{\bar{x}}, Z_{\bar{x}})$ are of sizes $p(k-1), pk, (p-1)k$ and $(p-1)k$, respectively.

Corollary 1.32. *If $\mathcal{K}_i\mathcal{S}_0 \cap \mathcal{G}$ and $\mathcal{K}_0\mathcal{S}_j \cap \mathcal{G}$ are polynomially determined for $i \leq k$ and $j \leq p$, then $\mathcal{K}_i\mathcal{S}_j \cap \mathcal{G}$ is polynomially determined for $i \leq k$ and $j \leq p$.*

If $\mathcal{G}$ is a class of graphs for which Min Coloring and Min Clique Cover are polynomially solvable, then for every fixed (i,j), $\mathcal{K}_i\mathcal{S}_0 \cap \mathcal{G}$ and $\mathcal{K}_0\mathcal{S}_j \cap \mathcal{G}$ are polynomially determined. This is for example the case if $\mathcal{G}$ is a subclass of perfect graphs.
A special case of this result is the test of 2-split-colorability [19].

Corollary 1.33. *($\mathcal{G}$ is the class of all finite graphs) Since $\mathcal{K}_i\mathcal{S}_0$ and $\mathcal{K}_0\mathcal{S}_j$ are polynomially determined for $i,j \leq 2$, $\mathcal{K}_2\mathcal{S}_2$ is also polynomially determined.*

The above algorithm is a nice generalization of the idea in [19] for recognizing $(1,2)$ and $(2,1)$-colorable graphs. Nevertheless, there is a simpler algorithm by Feder et al. giving the same result with a lower time complexity in [58]. Their result works in a more general framework where the objective is to partition the vertex set of a given graph into a sparse and a dense graph. In [58], $\mathcal{S}$ and $\mathcal{D}$ are defined as two classes of graphs, respectively called *sparse* and *dense*, satisfying the following conditions: both $\mathcal{S}$ and $\mathcal{D}$ are hereditary classes, and there exists a constant c such that the intersection $S \cap D$ has at most c vertices for any $S \in \mathcal{S}$ and $D \in \mathcal{D}$. A *sparse-dense partition* of a graph G with respect to the classes $\mathcal{S}$ and

$\mathcal{D}$, is a partition of the vertex set of G into two parts where one induces a sparse graph and the other one induces a dense graph. Their result is stated as follows.

Theorem 1.34 ([58]). *All sparse-dense partitions of a graph with n vertices can be found in time $\mathcal{O}(n^{2c+2}T(n))$, where $T(n)$ is the time for recognizing sparse and dense graphs.* □

Obviously, a (p,k)-coloring is a sparse-dense partition where the constant $c = pk$; a $(p,0)$-colorable graph is a dense graph and a $(0,k)$-colorable graph is a sparse graph. In this particular case, Theorem 1.34 can be restated as follows.

Corollary 1.35. *Let $\mathcal{G}$ be a hereditary class of graphs, let $(p,k) \in \mathbb{N} \times \mathbb{N}, p,k \geq 1$. If $\mathcal{K}_p\mathcal{S}_0 \cap \mathcal{G}$ (respectively $\mathcal{K}_0\mathcal{S}_k \cap \mathcal{G}$) is determined in time $\mathcal{O}(T_p(n))$ (respectively $\mathcal{O}(T_k(n))$), then $\mathcal{K}_p\mathcal{S}_k \cap \mathcal{G}$ can be determined in time $\mathcal{O}(n^{2kp+2}\max(T_p(n), T_k(n)))$.*

Their idea is based on the fact that a graph with n vertices has at most n^{2kp} different (p,k)-colorings. To show this, let $(\mathcal{K},\mathcal{S})$ be a particular (p,k)-coloring of G where $\mathcal{K}$ induces p cliques and $\mathcal{S}$ induces k stable sets. Then, any other (p,k)-coloring $(\mathcal{K}',\mathcal{S}')$ of G verifies $|\mathcal{K} \cap \mathcal{S}'| \leq kp$ and $|\mathcal{K}' \cap \mathcal{S}| \leq kp$. In other words, if we know a particular (p,k)-coloring then we can find all (p,k)-colorings by $2kp$-local searches. Then, the idea is to find a $(0,k)$-colorable induced subgraph as large as possible (in any possible (p,k)-coloring) by performing (at most n) $(2kp+1)$-local searches from a smaller $(0,k)$-colorable subgraph. This procedure ends with a $\mathcal{S}'$ having the same size as the unknown $\mathcal{S}$. The rest of the algorithm consists of performing a $2kp$-local search on $\mathcal{S}'$ until we find a (p,k)-coloring of G or, when the case fails to rise, we conclude that there is no such partition since we tested all possibilities.
The following result is then implied by both Theorem 1.31 and Corollary 1.35 since $(0,k)$-colorability and $(p,0)$-colorability are polynomially determined in the class of perfect graphs.

Corollary 1.36. *If $\mathcal{G}$ is a subclass of perfect graphs, then $\mathcal{K}_p\mathcal{S}_k \cap \mathcal{G}$ is polynomially determined for any fixed p and k.*

This corollary also follows from a result of [96] which states that, for any fixed p and k, hereditary families of (p,k)-colorable perfect graphs have a finite list of forbidden subgraphs, and hence a polynomial time recognition algorithm.
Another consequence of Theorem 1.34 concerns polar graphs.

Corollary 1.37. *For any graph G, and for fixed s and t, it can be recognized in polynomial time whether G is (s,t)-polar.*

Proof. First, note that a join of s stable sets is a sparse graph and that a union of t cliques is a dense graph. Then, one can observe that for fixed s and t, there can be at most $c = \min(s,t)$ vertices in the intersection of a $(s,0)$-polar graph and a $(0,t)$-polar graph. Furthermore, $(s,0)$-polar and $(0,t)$-polar graphs can be recognized in polynomial time; G is $(0,t)$-polar if and only if G does not contain induced P_3 and it has t connected components; G is $(s,0)$-polar if and only if $\overline{G}$ is $(0,s)$-polar. Note that the complexity is no more polynomial if s and t are not fixed. □

Let us note that the condition "$\mathcal{G}$ is hereditary" in Corollary 1.35 is needed unless $\mathcal{NP}$=$\mathcal{P}$. In fact, let us consider the class $\mathcal{G}$ of graphs obtained by adding a K_4 to an arbitrary graph G (without any edge between G and K_4). $\mathcal{G} \cap \mathcal{K}_0\mathcal{S}_3$ is trivially polynomially determined since such graphs are never $(0,3)$-colorable. On the other hand, $\mathcal{G} \cap \mathcal{K}_1\mathcal{S}_0$ is also polynomially determined (recognition of a clique). But it is $\mathcal{NP}$-complete to decide whether a graph of $\mathcal{G}$ is $(1,3)$-colorable. In fact a graph G is $(0,3)$-colorable if and only if the graph obtained by adding a K_4 to G, which is not connected to G, is $(1,3)$-colorable.
We observe that Corollaries 1.36 and 1.37 are also consequences of a result of Alekseev et al. in [3], which is obtained, independently from sparse-dense partitions, in the framework of recognizing $(\mathcal{P}_1, \ldots, \mathcal{P}_p, \mathcal{Q}_1, \ldots, \mathcal{Q}_k)$-colorable graphs (this notion is mentioned in Subsection 1.2.2).
In [40], we also point out that recognizing a $(1,3)$-coloring remains difficult even if a maximum induced (p,k)-colorable subgraph with (p,k) equal to $(1,0), (0,1), (0,2)$ and $(0,3)$ can be polynomially computed.

Proposition 1.38. *There exists a class of graphs $\mathcal{G}$ such that a maximum clique, a maximum stable set, a maximum induced bipartite and also a maximum 3-colorable induced subgraph can be polynomially computed on $\mathcal{G}$, while it is $\mathcal{NP}$-complete to decide whether a graph in $\mathcal{G}$ is $(1,3)$-colorable.*

Proof. Let us revisit the proof of $\mathcal{NP}$-completeness of graph 3-colorability by reduction to 3-SAT [71]. Given an instance of 3-SAT with p clauses $(C_1, \ldots, C_p)$ and n boolean variables $(x_1, \ldots, x_n)$, one constructs a graph $G = (V, E)$ defined by:

$$V = V_1 \cup V_2 \cup V_3,$$

where $V_1 = \{R, T, F\}$, $V_2 = \cup_{i=1}^{n} \{x_i, \bar{x}_i\}$ and $V_3 = \cup_{j=1}^{p} \{y_1^j, y_2^j, y_3^j, y_4^j, y_5^j\}$. V contains two vertices per literal, five vertices per clause and three color vertices R, T and F (standing for Red, True and False). Moreover,

$$E = E_1 \cup E_2 \cup E_3 \cup E_4,$$

where $E_1 = \{RT, RF, TF\}$, $E_2 = \cup_{i=1}^{n} \{Rx_i, R\bar{x}_i, x_i\bar{x}_i\}$ and $E_3 = \cup_{j=1}^{p} \{Ty_4^j,\ Ty_5^j,\ y_4^j y_5^j,\ y_4^j y_3^j,\ y_3^j y_2^j,\ y_3^j y_1^j,\ y_2^j y_1^j\}$. For every clause $C_i = (z_1, z_2, z_3)$, E_4 contains three edges linking the vertices of V_2 associated with (z_1, z_2, z_3) to (y_1^j, y_2^j, y_5^j), respectively.
G is 3-colorable if and only if the related 3-SAT instance is satisfiable. Moreover, denoting by Red, True and False the respective colors of (R, T, F) in this coloring, vertices in V_2 are either True or False, which defines a truth assignment to boolean variables satisfying every clause. It is also straightforward to verify that, for every 3-SAT instance, the related graph G admits a 4-coloring using the 4th color only for vertex T (in other words the graph obtained from G by removing vertex T is 3-colorable).
We denote by $r = 2n+5p+3$ the order of G, then we consider the graph $\tilde{G} = (\tilde{V}, \tilde{E})$ defined as follows: for every vertex v in V we introduce four copies $u_1^v, u_2^v, u_3^v, u_4^v$ in $\tilde{V}$. Finally, we add two vertices u'^T_4, u''^T_4 in $\tilde{V}$. Hence we have $\tilde{V} = \{u'^T_4, u''^T_4\} \cup_{v \in V} \{u_1^v, u_2^v, u_3^v, u_4^v\}$, while

$\tilde{E}$ is defined as follows: $\tilde{G}[\{u_1^v, v \in V\}], \tilde{G}[\{u_2^v, v \in V\}]$ and $\tilde{G}[\{u_3^v, v \in V\}]$ are three copies of G, $\tilde{G}[\{u_4^v, v \in V\} \cup \{u'^T_4, u''^T_4\}]$ is a $(r+2)$-clique, and finally, $\{u_1^v, u_2^v, u_3^v, u_4^v\}$ where $v \in V \setminus \{T\}$ and $\{u_1^T, u_2^T, u_3^T, u_4^T, u'^T_4, u''^T_4\}$ induce cliques in $\tilde{G}$.
By using a 3-coloring of $G[V \setminus \{T\}]$, it is easy to compute three disjoint stable sets S^1, S^2 and S^3 of $\tilde{G}$, each of cardinality r and containing respectively u_4^T, u'^T_4 and u''^T_4. Moreover, $\alpha(\tilde{G}) = r$ and consequently S^1 is a maximum stable set of $\tilde{G}$, $S^1 \cup S^2$ is a maximum induced bipartite subgraph of $\tilde{G}$, and $S^1 \cup S^2 \cup S^3$ is a maximum 3-colorable induced subgraph of $\tilde{G}$. On the other hand, $\{u_4^v, v \in V\} \cup \{u'^T_4, u''^T_4\}$ is the unique maximum clique of $\tilde{G}$. Consequently maximum stable set, maximum clique, maximum induced bipartite subgraph and maximum induced 3-colorable subgraph can be polynomially computed for every graph $\tilde{G}$ associated to 3-SAT instances. $\mathcal{G}$ is the class of such graphs. A graph in $\mathcal{G}$ is $(1,3)$-colorable if and only if the related 3-SAT instance is satisfiable, which is $\mathcal{NP}$-complete to verify. □

The proof of Proposition 1.38 suggests the following result.

Corollary 1.39. *There exists a class of graphs $\mathcal{G}$ such that a maximum clique, a maximum stable set, a maximum bipartite subgraph and a maximum tripartite subgraph can be polynomially computed on $\mathcal{G}$, whereas it is $\mathcal{NP}$-hard to find a maximum split graph.*

Proof. It suffices to remark in the proof of Proposition 1.38 that we have;

$$\alpha_{1,1}(\tilde{G}) = \begin{cases} r + (r+2) & \text{if } G \text{ is 3-colorable,} \\ r + (r+1) & \text{otherwise.} \end{cases}$$

This follows from the observation that $\tilde{G}$ admits a maximum stable set (of size r) which does not intersect with its unique maximum clique (of size $r+2$) if and only if G is 3-colorable. □

1.5 Min Split-coloring by integer programming

One may suggest the following integer programming model for Min Split-coloring in a given graph $G = (V, E)$ with $|V| = n$.

$$\begin{aligned}
\min \quad & \sum_j z_j & & \\
x_{ij} \quad &= \begin{cases} 1 & \text{if vertex } i \text{ is in the clique of split graph } j \\ 0 & \text{otherwise} \end{cases} & & (1.1)\\
y_{ij} \quad &= \begin{cases} 1 & \text{if vertex } i \text{ is in the stable set of split graph } j \\ 0 & \text{otherwise} \end{cases} & & (1.2)\\
x_{ij} + x_{kj} \quad &\leq 1 \text{ if } ik \notin E,\ j = 1, \ldots, l & & (1.3)\\
y_{ij} + y_{kj} \quad &\leq 1 \text{ if } ik \in E,\ j = 1, \ldots, l & & (1.4)\\
\sum_{i=1}^{n} (x_{ij} + y_{ij}) \quad &\leq n z_j,\ j = 1, \ldots, l & & (1.5)\\
\sum_{j=1}^{l} (x_{ij} + y_{ij}) \quad &= 1,\ i = 1, \ldots, n & & (1.6)\\
z_j \quad &\in \{0, 1\},\ j = 1, \ldots, l &&
\end{aligned}$$

Here l is an upper bound of $\chi_S(G)$. We may choose $l = \lceil n/3 \rceil$ since G is trivially $\lceil n/3 \rceil$-split-colorable by assigning one color to any triple of vertices.

Constraints (1.1) and (1.2) define the decision variables x_{ij} which give the vertices of G to be assigned to the clique of the split graph j, and y_{ij} the vertices in G to be assigned to the stable set of the split graph j. Constraints (1.3) express that no two vertices not linked in G can be in a same clique and, conversely, constraints (1.4) mean that no two vertices linked in G can be in a same stable set. Constraints (1.5) say that the number of vertices assigned to one split graph can not exceed n if this split graph exists, and 0 otherwise. Finally, constraints (1.6) make sure that every vertex is colored by exactly one color. The number of used colors is then given by the variables z_j; $z_j = 1$ means that color j is used in a solution and $z_j = 0$ says that the color j is not used.

One can notice that this model has $O(n^2)$ decision variables and $O(n^3)$ constraints. Advanced techniques have to be applied in order to solve this integer program for graphs having a large number of vertices. However its interest is theoretical and this formulation may be a basis for more efficient formulations involving additional constraints.

The above formulation suggests the following representation of Min Split-coloring: let k be a positive integer and let $1, \ldots, n$ be the vertices of G. We construct a graph $G(k)$ by first taking k copies $G^1, \ldots, G^k$ of G; let $\hat{y}_{ij}$ $(i = 1, \ldots, n)$ be the vertices of G^j. Then we take k copies $\overline{G}^1, \ldots, \overline{G}^k$ of $\overline{G}$ (the complement of G); let $\hat{x}_{ij}$ $(i = 1, \ldots, n)$ be the vertices of $\overline{G}^j$. Then for each i $(i = 1, \ldots, n)$, we form a clique on vertices $\hat{y}_{i1}, \ldots, \hat{y}_{ik}, \hat{x}_{i1}, \ldots, \hat{x}_{ik}$. The resulting graph is $G(k)$. We can then state the following proposition.

Proposition 1.40. *G has a k-split-coloring if and only if $G(k)$ has a stable set S with*

$|S| = n$.

Proof. There is a one-to-one correspondence between stable sets S with n vertices in $G(k)$ and k-split-colorings of G or, equivalently, integral solutions of the integer linear programming model (where k colors are used):

$$\hat{y}_{ij} \in S \Longleftrightarrow y_{ij} = 1;$$

$$\hat{x}_{ij} \in S \Longleftrightarrow x_{ij} = 1.$$

This means that $\hat{y}_{ij}$ is in S if and only if vertex i is in the stable set of the split graph j and, similarly, $\hat{x}_{ij}$ is in S if and only if vertex i is in the clique of the split graph j. □

Remark that if we simply consider $G(1)$ then one sees that there is a one-to-one correspondence between (maximum) split graphs in G and (maximum) stable sets in $G(1)$. It is interesting to notice that as soon as G contains an induced P_3, $G(1)$ is not perfect (it contains an induced C_5).
It is clear that solving Min Split-coloring by using the above integer programming model necessitates some elaborate techniques as for instance branch-and-cut. Development of such methods is not explored in this dissertation; the above integer programming model is only given in order to suggest new research fields in the study of Min Split-coloring. For instance, classes of graphs for which the integer programming formulation of Min Split-coloring has some nice properties which make it easy to solve can be studied.

Chapter 2

Cacti and triangulated graphs

In this chapter, we exhibit a class of graphs, called cacti, where Min Split-coloring is solved in polynomial time. In the first section, we derive polynomial time algorithms to solve Min Split-coloring and to find the split-independence number in cacti. These are based on the article [50].
In the second section, we mention similar results in triangulated graphs. Polynomial time algorithms for generalized coloring problems are easily derived from the results of Hell et al. [87], and the split-independence number follows from some structural properties of triangulated graphs.

2.1 Cacti

A *cactus* is defined as a connected simple graph where no two elementary cycles share an edge. In other words, no two elementary cycles have more than one common vertex. Let us remark that a tree is a cactus. Conversely, when each cycle of a cactus is contracted to a vertex, then we get a tree.
Note that in cacti, the size of a clique is at most three. For the split-coloring of cacti, we will focus on elementary odd cycles since the only connected components remaining after coloring odd cycles are even cycles, paths and trees, and once again, they are 2-colorable in the classical sense, i.e., 2-split-colorable by choosing only two stable sets. Moreover, we know that a cactus is bipartite if it contains no (elementary) odd cycles and hence it is 2-split-colorable. On the other hand, cacti are always 3-split-colorable because they are 3-colorable in the classical sense. In conclusion, the split-coloring of cacti boils down to be a decision between 2 and 3-split-colorability.
In the process of finding the split-independence number of a cactus, the assignment of a vertex to a stable set and/or to a clique associated with a split graph is strongly related to the structure (even or odd cycle, *bridge*, i.e., edge not contained in any cycle) to which this vertex belongs. Again, we will see that we have to proceed differently for vertices located in an odd cycle than for others.

For the reasons just mentioned, first, we need a procedure listing all odd cycles in a cactus. Although in arbitrary graphs, the number of odd cycles can grow exponentially with the number of vertices, their number is bounded above by $\lfloor (n-1)/2 \rfloor$ in cacti. This bound corresponds to the number of odd cycles in a cactus where a single central vertex is shared by all cycles of length three formed by the other $n-1$ vertices. In order to include odd cycles of a cactus $G = (V, E)$ with $|V| = n$, $|E| = m$ in a list $\mathcal{L}$, we first observe that any *2-connected component* (that is, a subgraph H of G which can be disconnected from G with the removal of at least two vertices) of a cactus is a cycle. Then we apply the algorithm of Tarjan [111] to detect all separating vertices and all cycles. This is a depth first search algorithm which provides a list of blocks and a list of separating vertices for any arbitrary graph in time linear in $\mathcal{O}(m)$. Note that a *separating vertex* is a vertex whose removal gives a disconnected graph, and a *block* is a maximal non-separable subgraph, so it may be either a 2-connected component or an *isthmus*, i.e., an edge whose removal increases the number of connected components [11]. Having a list of all blocks, it suffices to eliminate edges and cycles of even length to obtain the list $\mathcal{L}$ of odd cycles.

2.1.1 Split independence number in cacti

In this section, we concentrate on the problem of finding an induced split graph of maximum size in a cactus $G = (V, E)$, where $|V| = n$, $|E| = m$. In other words, we try to find a stable set S and a clique C in G such that $|S \cup C|$ is maximum. For this purpose, we will first describe a simple algorithm to find a maximum stable set in a cactus, since, to our knowledge, such an algorithm has not been given elsewhere.
We adopt a dynamic programming approach. Given a cactus G and its lists of cycles and separating vertices, one can construct a corresponding arborescence A using the block-cutpoint tree (see [83]):

- each cycle, each edge not contained in a cycle and each separating vertex in G is represented by a vertex in A;
- two vertices are linked in A if and only if they correspond to a block and a separating vertex contained in it;
- all the edges in A are oriented towards a chosen separating vertex (preferably one closest to a leaf).

Note that vertices corresponding to two cycles, or two separating vertices in G are not linked in A. Having such an arborescence A, one can label separating vertices (v_j) in increasing order according to the direction of exploration of A. Then, another label may be given to vertices representing cycles in such a way that labels of cycles linked to and oriented towards v_j would be smaller than the labels of cycles linked to and oriented towards v_i whenever $j < i$. An example of labeling can be seen in Figure 2.1 where the labels of cycles, edges and separating vertices (black ones) of G are determined by means of the arborescence A. This

way of labeling G is not unique but perfectly defines a direction for exploring the original graph G by dynamic programming.

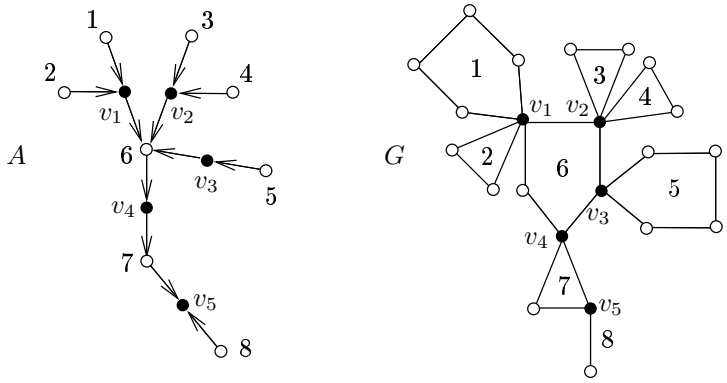


Figure 2.1: An arborescence A corresponding to graph G.

One may proceed in the exploration order of separating vertices and compute the stability number in the subgraph G_j lying in the lower part of v_j (which consists of cycle(s) oriented towards v_j) according to two scenarios: v_j is excluded in the maximum stable set of G_j or it is authorized. Hence, for any separating vertex v_j, we will compute two weights; $w^e(v_j) =$ maximum weight of a stable set in $G_j - v_j$, and $w^a(v_j) =$ maximum weight of a stable set containing v_j in G_j, where the weight of a stable set is simply the sum of the weights of its vertices. Having computed $w^e(v_j)$ and $w^a(v_j)$ for a separating vertex, we can replace $G_j - v_j$ by one vertex $v(G_j)$ of weight $w^e(v_j)$ since its best contribution to a maximum stable set of G, which, in this case, does not contain v_j, is equal to $w^e(v_j)$. On the other hand, putting v_j in a stable set will imply that we add $w^a(v_j)$ vertices in a maximum stable set of G, and that $v(G_j)$ cannot be put in the same stable set. Consequently, the weight of the vertex v_j will be equal to $w^a(v_j)$.

One can explore a whole cactus G repeating this procedure until we obtain a maximum stable set of G. Note that both for computing $w^e(v_j)$ and $w^a(v_j)$, the problem with which we have to deal is exactly the maximum weighted stable set problem in a *caterpillar* (i.e., a tree with the property that the removal of its leaves results in a path); this is easily solved. Once an optimal solution is obtained, this maximum weight contains, as information, the set of vertices giving the corresponding maximum stable set. The number of calculations of this kind is equal to the number of separating vertices. Algorithm 2 describes the application of the above idea in order to find a maximum stable set in a cactus. In Figure 2.2, the method is represented for the graph G of Figure 2.1. At each step, only the weights that we need are shown in brackets. Note that the remaining graph after the application of Algorithm 2 is either a path or a cycle with an edge pending from one of its vertices; therefore the final computation of the algorithm is trivial.

The above discussion leads to the following result.

Lemma 2.1. *For a cactus G, $\alpha(G)$ can be found in time linear in $\mathcal{O}(m+k)$, where k is the number of separating vertices in G.*

Algorithm 2 Maximum stable set in cacti

Input: a cactus G

Output: a maximum stable set of G

1. set the weights of all vertices to 1;
2. determine separating vertices and the list $\mathcal{L}$ of odd cycles in G;
3. construct an arborescence A;
4. **for all** separating vertex v_j (considered in the exploration order of A) **do**
5. compute $w^e(v_j)$ = weight of a maximum weighted stable set of G_j not containing v_j;
6. compute $w^a(v_j)$ = weight of a maximum weighted stable set of G_j containing v_j;
7. replace $G_j - v_j$ by a vertex $v(G_j)$ of weight $w^e(v_j)$, assign the weight $w^a(v_j)$ to v_j, link $v(G_j)$ to v_j;
8. compute the weighted stability number of the remaining graph and return the corresponding stable set.

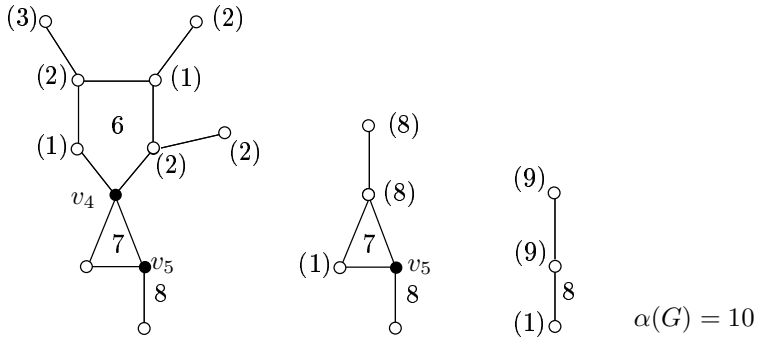


Figure 2.2: Computing the stability number of a cactus G.

Theorem 2.2. *For a cactus G, an induced split graph of maximum size can be found in linear time.*

Proof. First of all, Algorithm 2 returns a maximum stable set S_{max} giving $\alpha(G)$. Observe that we have $\alpha(G) + \omega(G) - 1 \leq \alpha_S(G) \leq \alpha(G) + \omega(G)$ for any graph. Having the list $\mathcal{L}$ of all odd cycles, we search for a triangle K_3^i in it which verifies $\alpha(G - K_3^i) = \alpha(G)$ for a maximum stable set S_{max}^i of $G - K_3^i$. If such a pair (K_3^i, S_{max}^i) exists then it constitutes a split graph of maximum cardinality $\alpha_S(G) = \alpha(G) + 3$ in G. Otherwise any pair (K_3^i, S_{max}^i) is a split graph of maximum cardinality $\alpha_S(G) = \alpha(G) + 2$. Whenever $\mathcal{L}$ contains no K_3, a clique of size 2 can be chosen in such a way that none of its vertices is contained in a maximum stable set S_{max} of G. This claim is true since there would necessarily be two adjacent vertices of an odd cycle which do not appear in S_{max}. Finally, if $\mathcal{L}$ contains no odd cycles then obviously $\alpha_S(G) = \alpha(G) + 1$. □

2.1.2 Split-coloring of cacti

Having the list of all odd cycles of G, we will consider an auxiliary graph $G' = (V', E')$ where every vertex $v'_i \in V'$ corresponds to an odd cycle OC^i of G, hence $|V'| = |\mathcal{L}| = L$. Two vertices v'_i and v'_j of V' are linked if and only if the distance between OC^i and OC^j is at most 1, i.e., either OC^i and OC^j have a common vertex or there are vertices in OC^i and OC^j which are adjacent. Note that if G is a cactus then so is G' since, by construction, two cycles sharing an edge in G' implies that there are cycles sharing at least one edge in G.

Algorithm 3 Split-coloring of cacti

Input: a cactus G

Output: $\chi_S(G)$

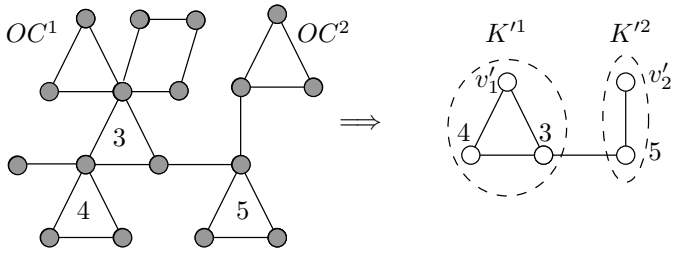
1. construct G'; *//as already described in the proof of Theorem 2.3*
2. **if** there is a stable set of size three in G' **then**
3. **print** "$\chi_S(G) = 3$", **exit**.
4. **else**
5. choose v'_1 and v'_2 such that $v'_1v'_2 \notin E'$, set $K'^1 = \{v'_1\}$, $K'^2 = \{v'_2\}$ and $R = \emptyset$;
6. **for all** vertex $v'_i \in V'$ **do**
7. **if** $v'_iv'_1 \in E'$ and $v'_iv'_2 \notin E'$ **then**
8. store v'_i in K'^1;
9. **else if** $v'_iv'_2 \in E'$ and $v'_iv'_1 \notin E'$ **then**
10. store v'_i in K'^2;
11. **else**
12. store v'_i in R;
13. **for all** vertex $v'_i \in R$ **do**
14. **if** v'_i is adjacent to every vertex in K'^1 (respectively in K'^2) **then**
15. remove v'_i from R to K'^1 (respectively to K'^2);
16. **if** $R = \emptyset$ **then**
17. K'^1 and K'^2 are both cliques;
18. **print** "$\chi_S(G) = 2$", **exit**.
19. **else**
20. $\exists v \in V'$ which constitutes a clique neither with K'^1 nor with K'^2;
21. **print** "G is a OC_5-cactus and $\chi_S(G) = 3$", **exit**.

Theorem 2.3. *Algorithm 3 decides in $\mathcal{O}(L^3)$ time whether a cactus is 2 or 3-split-colorable.*

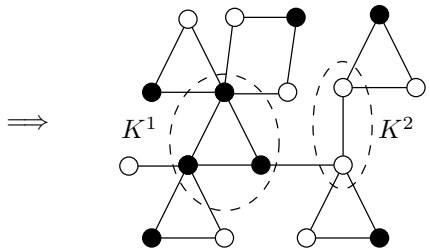
Proof. We first remark that the stability number of G' is the largest number of induced disjoint union of odd cycles in G. More precisely, we are not interested in the exact value of $\alpha(G')$ because we know that if $\alpha(G') \geq 3$ then $\chi_S(G) \geq 3$, due to Fact 1.26. Therefore, a simple procedure which tests for every triplet in G', in $\mathcal{O}(L^3)$ time, whether it forms a stable set or not does the job. Having such a triplet, we may claim that our cactus is

3-split-colorable.

Otherwise we fix an arbitrary pair of non-adjacent vertices $\{v'_1, v'_2\}$ corresponding to two induced odd cycles in G, and we try to partition the vertex set V' into two cliques K'^1 and K'^2 containing respectively v'_1 and v'_2. One can observe that having two cliques covering the vertex set V' of G', we can trivially exhibit a 2-split-coloring of G. More precisely, the existence of a clique in G' implies that there is a set of odd cycles in G which are pairwise at distance at most 1, and that there is a clique in G of size 1, 2 or 3, which contains at least one vertex of each odd cycle of this set. In other words, the two cliques K'^1 and K'^2 in G' found by Algorithm 3 tell us exactly how to choose two cliques K^1 and K^2 in G, each one associated with a split graph covering together the vertex set of G according to Fact 1.24. That is why Algorithm 3 searches for a partition into K'^1 and K'^2 that we would like to be cliques. Therefore, K'^1 (respectively K'^2) contains only vertices which are adjacent to v'_1 (respectively v'_2). We can see in Figure 2.3 an illustration of how Algorithm 3 gives a 2-split-coloring of the graph given in a).



a) A cactus G. b) G' partitioned into K'^1 and K'^2.

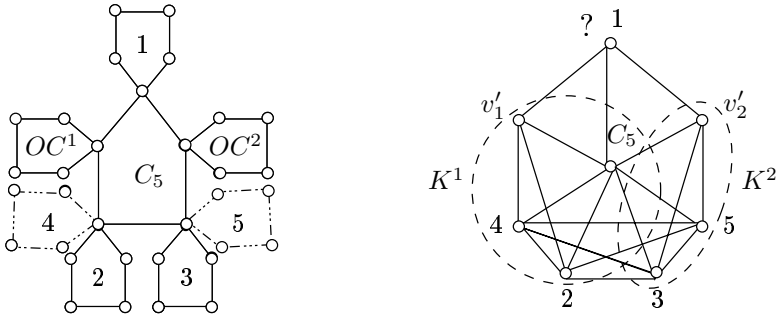


c) 2-split-coloring of G.

Figure 2.3: An example of split-coloring of a cactus.

In general, the phase of vertex assignment yields two cliques K'^1 and K'^2 because of the fact that we have at most two induced odd cycles. There is only one case which is easily detected by Algorithm 3, where we do not obtain this result. It is due to the possible type of connection between two fixed induced odd cycles OC^1 and OC^2: OC^1 and OC^2 linked by a C_5. This exception, called C_5-*cactus*, consists in a central cycle of length 5 sharing each one of its vertices with at least one other odd cycle; an example of C_5-cactus appears in Figure 2.4. We observe that there is no way to partition the vertices of the auxiliary graph of a C_5-cactus into two cliques.

It follows that a cactus G is 2-split-colorable if and only if there are at most two induced



a) A C_5-cactus. b) Its auxiliary graph.

Figure 2.4: C_5-cacti are 3-split-colorable.

odd cycles and G is not a C_5-cactus; it will be 3-split-colorable otherwise. □

Here we find an other 3-split-critical structure in addition to the ones given in Chapter 1, Section 1.3: a C_5-cactus with unique odd cycles linked to each vertex of the central C_5 is 3-split-critical.

Finally, let us notice that for Min Cocoloring, we have obviously $z(G) \leq 3$ for any cactus G by taking a usual coloring. Moreover, we have $z(G) = 2$ if and only if either $\chi(G) = 2$, or $\theta(G) = 2$, or it is a split-graph. Note that all of these cases are trivially detected. One can also easily see that general (p, k)-coloring problems can be solved by methods similar to the one of Min Split-coloring. Nevertheless, they do not seem to be interesting to develop because of the chromatic number of cacti which is bounded above by three.

2.2 Triangulated graphs

Recall that a graph G is triangulated if any induced cycle C_k of G with $k \geq 4$ has a chord. In what follows, we exhibit the results on all (p, k)-coloring problems in triangulated graphs. First, let us introduce some notions that will be used in the sequel.

Definition 2.4 (Simplicial vertex). *Given a graph $G = (V, E)$, a vertex $v \in V$ is called simplicial if $N(v)$ is a clique in G.*

Definition 2.5 (Perfect elimination order). *Given a graph G, an order $(1, 2, \ldots, n)$ is a perfect elimination order if for all $i, N_i = \{j : j \in N(i)$ and $j > i\}$ is a clique, i.e., i is a simplicial vertex in the graph induced by vertices $i + 1, \ldots, n$.*

It is known that a graph G is triangulated if and only if it admits a perfect elimination order [46, 67]. Moreover, given a triangulated graph G, one can find $\chi(G)$ and all maximal cliques of G in time $\mathcal{O}(n + m)$ [112] and also $\alpha(G)$ in the same time complexity [72], all with algorithms based on a perfect elimination order. Finally, note that an optimal coloring is obtained by a *reverse greedy algorithm* which proceeds in the order $n, n - 1, \ldots, 1$ and assigns to each vertex the smallest available color.

2.2.1 Split-independence number in triangulated graphs

Theorem 2.6. *For a triangulated graph $G = (V, E)$, the split-independence number $\alpha_S(G)$ can be obtained in $\mathcal{O}(D(n+m))$ time, where D is the number of maximum cliques in G.*

Proof. It is easily seen that a maximum stable set and a maximum clique cannot share more than one vertex, hence $\alpha(G) + \omega(G) - 1 \leq \alpha_S(G) \leq \alpha(G) + \omega(G)$. Having the list of all maximum cliques K^i's, obtained from the list of all maximal cliques by eliminating the ones which are not maximum, it suffices to find a maximum stable set S^i in $G - K^i$. It should be trivially noted that $\alpha(G) - 1 \leq |S^i| \leq \alpha(G)$. If there exists S^i such that $|S^i| = \alpha(G)$ for some i, then S^i is also a maximum stable set in G and $K^i \cup S^i$ is a maximum split graph of size $\alpha_S(G) = \alpha(G) + \omega(G)$. Otherwise, we can conclude that there is no pair of maximum clique and maximum stable set having no common vertex. In this case, any pair (K^i, S^i) forms a maximum split graph of size $\alpha_S(G) = \alpha(G) + \omega(G) - 1$. To decide, we only run once the algorithm finding all maximal cliques in G, and D times the algorithm finding a maximum stable set in $G - K^i$, where D is the number of maximum cliques. □

Let us note that, in triangulated graphs, it is rather hard to generalize this result to find $\alpha_{p,k}$ in polynomial time for given p and k; in fact, it is already $\mathcal{NP}$-hard to find $\alpha_{p,0}$ and $\alpha_{0,k}$ in split graphs which are triangulated and co-triangulated (i.e., their complement is triangulated) [120]. In return, this problem becomes interesting for fixed p and k since both finding $\alpha_{p,0}$ and $\alpha_{0,k}$ are polynomially solvable in triangulated graphs for fixed p and k [120]. This direction can be explored in a future research.

2.2.2 (p, k)-coloring of triangulated graphs

In this section, we derive polynomial time algorithms for all (p, k)-coloring problems based on the results of Hell et al. [87]. The following theorem characterizes (p, k)-colorable triangulated graphs by forbidden subgraphs.

Theorem 2.7 ([87]). *A triangulated graph is (p, k)-colorable if and only if it does not contain $(p+1)K_{k+1}$ as an induced subgraph.* □

Note that if an arbitrary graph contains $(p+1)K_{k+1}$ as an induced subgraph then it is trivially not (p, k)-colorable. Nevertheless, the converse is true only for triangulated graphs. Algorithm 4 is then derived in [87] for the recognition of (p, k)-colorable triangulated graphs. Note that c_j and s_j are the colors corresponding respectively to the j'th clique and the j'th stable set used to color the vertices of a given graph. We denote by S^i the set consisting of i, together with all vertices among $1, 2, \ldots, i-1$ colored $s_1, s_2, \ldots, s_p$.

Corollary 2.8. *(p, k)-coloring problems, Min Split-coloring and Min Cocoloring are polynomially solvable in triangulated graphs.*

Algorithm 4 Recognition of (p,k)-colorable triangulated graphs [87]

Input: a triangulated graph G with a perfect elimination ordering $1,2,\ldots,n$ of its vertices, an integer p

Output: a (p,k_{min})-coloring of G

1. color the vertex 1 by s_1;
2. **if** vertices $1,2,\ldots,i-1$ are colored without using color c_1 **then**
3. **if** $G[\{1,\ldots,i\}]$ is p-colorable **then**
4. remove the colors from $1,2,\ldots,i-1$ and color $1,2,\ldots,i$ with colors $s_1,s_2,\ldots,s_p$; *//using the reverse greedy algorithm*
5. **else**
6. keep the coloring of $1,2,\ldots,i-1$ and color i with c_1;
7. **if** vertices $1,2,\ldots,i-1$ are colored having used colors $c_1,c_2,\ldots,c_a$ **then**
8. **if** $\exists b \leq a$ which is the least subscript such that i is adjacent to the first vertex colored with c_b **then**
9. color i with c_b;
10. **else if** S^i is p-colorable **then**
11. remove the colors from the vertices of $S^i \setminus i$ and color S^i with colors $s_1,s_2,\ldots,s_p$; *//using the reverse greedy algorithm*
12. **else**
13. keep the coloring of $1,2,\ldots,i-1$ and color i with c_{a+1}.

Proof. Although Algorithm 4 is conceived to recognize a (p,k)-colorable triangulated graph, in fact, it minimizes k such that the graph is (p,k)-colorable for any given p; we set k_{min} to be the largest value of a, such that there is a vertex colored c_a, or $k_{min}=0$ if all vertices are colored with $s_1,s_2,\ldots,s_p$. Furthermore, it is done in time $\mathcal{O}(n(n+m))$ since it works with at most n applications of reverse greedy algorithm. The solutions for (p_{min},k)-coloring, Min Split-coloring and Min Cocoloring in triangulated graphs can be then computed in time $\mathcal{O}(n^2(n+m))$ according to Remarks 1.20 and 1.21. □

An algorithm with better time complexity, namely $\mathcal{O}(n(n+m))$, for Min Cocoloring is derived in [78]. It explores the characterization of triangulated graphs which states that they are exactly the intersection graphs of subtrees in a tree [73]. The question of whether Min Split-coloring admits an algorithm with time complexity better than $\mathcal{O}(n^2(n+m))$ in triangulated graphs is an interesting question that we leave for future research. Also, a natural continuation of these results would be to consider the generalized coloring problems in classes of graphs containing triangulated graphs, like quasi-triangulated graphs[1] and weakly triangulated graphs[2].

Note finally that Hell et al. give also a simplification of Algorithm 4 for $p=k=1$ in [87].

[1] A graph G is *quasi-triangulated* if it admits an order of its vertices such that for any subgraph H of G, either the largest vertex in H is simplicial in H or the smallest vertex in H is simplicial in $\overline{H}$.

[2] A graph G is *weakly triangulated* if neither G nor $\overline{G}$ contains chordless cycles of length greeter than or equal to five.

This allows to derive a recognition algorithm of split graphs in time $\mathcal{O}(n+m)$, which is a better time complexity than the algorithm in time $\mathcal{O}(n^2)$ given in [100].

Chapter 3

Cographs

In this chapter which exhibits the results obtained in [40], we focus on a subclass of perfect graphs: the cographs. Cographs are intensively studied in the literature with respect to various partitioning problems. Thanks to their nice structural properties (we introduce them in Section 3.1 that follows), they generally admit polynomial time algorithms for problems (including Min Coloring, Min Clique Cover, Max Stable Set and Max Clique) which are not always that easily solvable in other classes of graphs. For instance, precoloring extension can be solved in linear time in cographs [92] while it is $\mathcal{NP}$-complete in bipartite graphs [90]. In a similar way, there is a linear time algorithm solving chromatic sum in cographs while the same problem is $\mathcal{NP}$-complete in the class of bipartite graphs [91]. But, contrarily to this first impression, there is also a significant number of $\mathcal{NP}$-complete problems in cographs: for instance achromatic number[1] [12], list coloring [92], partition into three stable sets of size at most l [13] and generalized chromatic sum [91]. In this context, the complexities of generalized coloring problems appear as an open problem far from being trivial.
In this chapter, we give a new characterization of cographs which leads to efficient algorithms for solving (p,k)-coloring problems, Min Split-coloring and Min Cocoloring respectively in $\mathcal{O}(n^2+nm)$ time, $\mathcal{O}(n^2+nm)$ time and $\mathcal{O}(n^{3/2})$ time. Here, we improve the time bound given in [78] for Min Cocoloring of cographs by answering an open question of [65]: we show that the greedy cocoloring algorithm given in [65] is an exact algorithm only for cographs. We also give a dynamic programming algorithm to find the maximum induced (p,k)-colorable subgraph. In addition, we present characterizations of $(2,1)$- and $(2,2)$-colorable cographs by forbidden configurations[2]. Damaschke proves in [34] that there cannot be infinitely many cographs without having one as an induced subgraph of another. This implies that any hereditary subfamily (e.g. (p,k)-colorable cographs or polar cographs) must be characterized by a finite set of forbidden induced subgraphs. Finally, we also mention some results of Feder et al. [57] concerning different generalized colorings of cographs and

[1] Given a graph $G=(V,E)$, its achromatic number is the maximum number of stable sets $V_1,\ldots,V_k$ partitioning V and such that, for each pair of distinct sets V_i and V_j, $V_i \cup V_j$ is not a stable set.

[2] These ideas have been used since then by Francisco et al. in [66] to characterize (p,k)-colorable cographs, for all p and k.

determining in particular (p,k)-colorability of cographs in linear time for fixed p and k.

3.1 Preliminaries on cographs

We will consider *cographs,* also called P_4-free graphs, which are defined as graphs without induced path on four vertices, i.e., without P_4.
Cographs were first introduced as complement reducible graphs by Corneil et al. in [29]. Note that the complement of a cograph is a cograph as well since $\overline{P_4}$ is isomorphic to a P_4. They are a subclass of perfect graphs admitting linear time algorithms for Min Coloring, Min Clique Cover, Max Stable Set and Max Clique [30]. In fact, it is shown in [25] that, given a graph, while applying the greedy coloring algorithm assigning the first available color to its vertices considered in some order, the only structure preventing the algorithm to give an optimal solution occurs on a particular ordering of vertices of an induced P_4. Since there is no induced P_4 in cographs, a greedy coloring algorithm gives an optimal coloring for all orderings of vertices; this is obtained in linear time in the size of the graph.
It is well-known that for a cograph G, either G or $\overline{G}$ is disconnected (see [29]). Furthermore, cographs are precisely the graphs that can be constructed from one vertex using the operations of taking disjoint unions and taking complements. Subsequently, a unique tree can be constructed with cograph G as the root and with internal vertices labeled to mark which of these two operations joins subgraphs. This tree is known as the *cotree* and can be constructed in time $\mathcal{O}(n+m)$ [29]. In the cotree $T=(N,A)$ associated with a cograph G, every leaf corresponds to a vertex and vice versa. Every internal vertex x is labeled by its type $t(x)$ (either 0 or 1) and corresponds to the subgraph $G(x)$ of G containing all children of x. If $t(x)=0$ (i.e., x is a *0-vertex*), then $G(x)$ is disconnected and every connected component induces a child of x. If $t(x)=1$ (i.e., x is a *1-vertex*), then $G(x)$ is connected and every connected component of $\overline{G(x)}$ induces a child of x. If x denotes a vertex in the cotree, then $c_1x, c_2x, \ldots$ are the children of x and $C(x)$ is the set of its children. Given the cotree of a cograph, a maximum clique and a maximum stable set can be found in time $\mathcal{O}(n)$.
Let us finally remark that in cographs, induced split graphs reduce to *threshold graphs*; these are defined as a subclass of split graphs for which there is an ordering of vertices such that the neighborhoods form a nested family [100]. In other words, a threshold graph admits a split partition (S,K) where for any two vertices v,u in S, the sets of neighbors satisfy $N(v) \supseteq N(u)$ or $N(u) \supseteq N(v)$. In fact, threshold graphs are exactly $(2K_2, C_4, P_4)$-free graphs [26].

3.2 Max (p,k)-colorable subgraph in cographs

Theorem 3.1. *A maximum induced (p,k)-colorable subgraph can be computed in time $\mathcal{O}((p^3k+pk^3)n)$ in cographs defined by their cotree.*

Proof. The algorithm which is based on dynamic programming moves from the leaves to the root of the cotree and computes the matrix $M(x) = (M_{i,j}(x))_{(i,j)\in\ \{0,\ldots,p\}\times\{0,\ldots,k\}}$ for every vertex x, where $M_{i,j}(x)$ is a maximum (i,j)-colorable subgraph of $G(x)$ (by convention $M_{0,0}(x) = \emptyset$).
Let us point out that, given a graph G with h connected components $G_1, \ldots, G_h$, $\alpha_{i,j}(G)$ is the optimal value of the following problem (with variables $(i_1, \ldots, i_h)$):

$$P_{\alpha_{i,j}} : \begin{cases} \max & \sum_{l=1}^{h} \alpha_{i_l,j}(G_l) \\ s.t. & \sum_{l=1}^{h} i_l = i \\ & i_l \in \{0,\ldots,i\} \end{cases}$$

Moreover, if $(i_1, \ldots, i_h)$ is an optimal solution, then $G[V_1 \cup \ldots \cup V_h]$, where $G_l[V_l]$ is a maximum (i_l, j)-colorable subgraph of G_l, is a maximum (i,j)-colorable subgraph of G.
An optimal solution of $P_{\alpha_{i,j}}$ can be computed in time $\mathcal{O}(i^2 h)$ by dynamic programming from the matrix $(\alpha_{i',j}(G_{h'}))_{(i',h')\in\{0,\ldots,i\}\times\{1,\ldots,h\}}$.
This leads to Algorithm 5 below based on two procedures called respectively Compose and Decompose. Procedure Compose computes the matrix $M(x)$ from matrices $M(cx)$, where $cx \in C(x)$, by solving problems of type $P_{\alpha_{i,j}}$. For every i,j, a maximum (i,j)-colorable subgraph $M_{i,j}(x)$ of $G(x)$ is computed by dynamic programming: $\tilde{M}^r_{s,j}$ (respectively $\check{M}^r_{i,s}$) denotes a maximum (s,j)-colorable subgraph of the graph induced by the first r connected components of $G(x)$ (respectively a maximum (i,s)-colorable subgraph of the graph induced by the first r connected components of $\overline{G(x)}$). Procedure Decompose is a recursive procedure which uses the previous one and moves from the leaves to the root according to the dynamic programming principle.

Algorithm 5 Maximum (p,k)-colorable subgraph in cographs

Input: a cograph G and the corresponding cotree $T = (N, A)$ rooted in x_0, two integers p and k

Output: a maximum (p,k)-colorable subgraph of G

1. Decompose(x_0, p, k);
2. **return** $M_{p,k}(x_0)$.

A cograph G and the related cotree $T = (N, A)$ are given as well as two integers p and k. The execution of Procedure Compose for a vertex x takes time $\mathcal{O}(p^3 k|C(x)|)$ if $t(x) = 0$, and $\mathcal{O}(pk^3|C(x)|)$ if $t(x) = 1$. Let us finally notice that every internal vertex of a cotree has at least two children; consequently, the number of internal vertices is at most $F - 1$ where F is the number of leaves; moreover, in the cotree associated with G, $F = n$ is the size of G. Consequently, the whole complexity is $\mathcal{O}((p^3 k + pk^3)n)$, which concludes the proof. □

Procedure 6 Compose

Input: $x \in N$, $(M(cx), cx \in C(x) = \{c_1 x, \ldots, c_{|C(x)|}x\})$, two integers p and k

Output: $M(x)$

1. **if** $t(x) = 0$ **then**
2. **for all** $(i,j) \in \{0,\ldots,p\} \times \{0,\ldots,k\}$ **do**
3. **for** $s \leftarrow 0$ to i **do**
4. $\tilde{M}^1_{s,j} \leftarrow M_{s,j}(c_1 x)$;
5. **for** $r \leftarrow 2$ to $|C(x)|$ **do**
6. **for** $s \leftarrow 0$ to i **do**
7. $q_0 = \text{argmax}\big(|\tilde{M}^{r-1}_{q,j}| + |M_{s-q,j}(c_r x)|, q \in \{0,\ldots,s\}\big)$;
8. $\tilde{M}^r_{s,j} \leftarrow \tilde{M}^{r-1}_{q_0,j} \cup M_{s-q_0,j}(c_r x)$;
9. $M_{i,j}(x) \leftarrow \tilde{M}^{|C(x)|}_{i,j}$;
10. **else** *//t(x)=1*
11. **for all** $(i,j) \in \{0,\ldots,p\} \times \{0,\ldots,k\}$ **do**
12. **for** $s \leftarrow 0$ to j **do**
13. $\check{M}^1_{i,s} \leftarrow M_{i,s}(c_1 x)$;
14. **for** $r \leftarrow 2$ to $|C(x)|$ **do**
15. **for** $s \leftarrow 0$ to j **do**
16. $q_0 = \text{argmax}\big(|\check{M}^{r-1}_{i,q}| + |M_{i,s-q}(c_r x)|, q \in \{0,\ldots,s\}\big)$;
17. $\check{M}^r_{i,s} \leftarrow \check{M}^{r-1}_{i,q_0} \cup M_{i,s-q_0}(c_r x)$;
18. $M_{i,j}(x) \leftarrow \check{M}^{|C(x)|}_{i,j}$.

Procedure 7 Decompose

Input: $x \in N$, two integers p and k

Output: $M(x)$

1. **if** $C(x) = \emptyset$ **then**
2. **for all** $(i,j) \in \{0,\ldots,p\} \times \{0,\ldots,k\}$ **do**
3. **if** $i + j = 0$ **then**
4. $M_{i,j}(x) \leftarrow \emptyset$;
5. **else**
6. $M_{i,j}(x) \leftarrow \{x\}$;
7. **else**
8. **for all** $y \in C(x)$ **do**
9. Decompose(y);
10. Compose$(x, (M(cx), cx \in C(x)), p, k)$.

3.3 (p,k)-coloring of cographs

Here is the main result of this chapter which suggests a new characterization of cographs (*Statement 3* of the following theorem) using their (p,k)-colorability.

Theorem 3.2. *For any graph $G=(V,E)$, the following statements are equivalent:*

1. *G is a cograph;*
2. *$\forall V' \subseteq V,\ \alpha_{1,1}(G[V']) = \alpha(G[V']) + \omega(G[V']) - 1$;*
3. *$\forall V' \subseteq V$ such that $G[V'] \in \mathcal{K}_p\mathcal{S}_k$, where $p \geq 1$, let K be a maximum clique of $G[V']$, then $G[V' \backslash K] \in \mathcal{K}_{p-1}\mathcal{S}_k$.*

Proof. 1. $\Rightarrow$ 2. It is shown in [29] that any cograph G has the clique-kernel intersection property, which means that every maximal clique of G has one vertex in common with every maximal stable set of G. Thus, there is no disjoint pair of a maximum clique and a maximum stable set in a cograph and, obviously, this property is hereditary.
1. $\Leftarrow$ 2. Assume that G is not a cograph, then it will contain at least one induced P_4. It suffices to observe that $\alpha_{1,1}(P_4) = \alpha(P_4) + \omega(P_4) = 2 + 2 = 4$ implying that *Statement 2* does not hold.
3. $\Rightarrow$ 1. It suffices to notice that a P_4 with edges a_1a_2, a_2a_3 and a_3a_4 belongs to $\mathcal{K}_1\mathcal{S}_1$ but taking a_1a_2 as a maximum clique does not yield a $(0,1)$-colorable graph.
1. $\Rightarrow$ 3. Let $(K^1, \ldots, K^p, S^1, \ldots, S^k)$ be a partition of V into p cliques and k stable sets. One can decompose any maximum clique K into two parts K_C and K_S, where $K_C = K \cap (K^1 \cup \ldots \cup K^p)$ and $K_S = K \cap (S^1 \cup \ldots \cup S^k)$. Obviously, K_S has at most k vertices. Without loss of generality, one can write $K_S = \{x_1, \ldots, x_s\}$, where $x_i \in S^i$ and $s \leq k$. An illustration of the proof is given in Figure 3.1 where the clique K is shown by light shadowed sets and where the dotted lines describe stable sets.

On the other hand, there is a vertex set $L \subseteq K^1 \cup \ldots \cup K^p$ with $|L| \leq |K_S| = s$ such that $K_C \cup L$ is a maximal clique of $K^1 \cup \ldots \cup K^p$. Since G is a cograph, we can immediately say that $(K^1 \cup \ldots \cup K^p) \backslash (K_C \cup L) \in \mathcal{K}_{p-1}\mathcal{S}_0$ by coloring the complementary graph. Then, knowing that $(S^{s+1} \cup \ldots \cup S^k) \in \mathcal{K}_0\mathcal{S}_{k-s}$, it suffices to show that $L \cup \left[(S^1 \cup \ldots \cup S^s) \backslash K_S\right] \in \mathcal{K}_0\mathcal{S}_s$ to obtain the assertion, namely $G[V \backslash K] \in \mathcal{K}_{p-1}\mathcal{S}_k$.
Now, let us suppose that there is a clique K' of size $s+1$ in $(L \cup [(S^1 \cup \ldots \cup S^s) \backslash K_S]$. Clearly $K' \cap L \neq \emptyset$ since $S^1 \cup \ldots \cup S^s \in \mathcal{K}_0\mathcal{S}_s$ and $|K'| = s+1$. In the same way, we know that $K' \cap \left[(S^1 \cup \ldots \cup S^s) \backslash K_S\right] \neq \emptyset$ because $|K'| > |L|$. In what follows, we define $K'_L = K' \cap L = \{z_1, \ldots, z_{l'}\}$ and $K'_S = K' \cap [(S^1 \cup \ldots \cup S^s) \backslash K_S] = \{a_1, \ldots, a_h, a_{h+1}, \ldots, a_{s+1-l'}\}$ such that $\forall i \leq h,\ \{a_i\} \cup K_C$ is a clique (a_i's are possibly re-indexed) and that $\forall i \geq h+1,\ \exists w_i \in K_C$ such that $a_i w_i \notin E$ (possibly $h = s+1-l'$). K' is shown as dark shadowed sets in Figure 3.1. Now, let us observe the three following facts which will allow us to conclude.

Fact 3.3. *$\{a_j, z_i, w_j, x_j\}$ does not induce a $P_4 \Rightarrow \forall h,\ h+1 \leq j \leq s+1-l',\ \forall i \leq l',\ x_j z_i \in E$.*

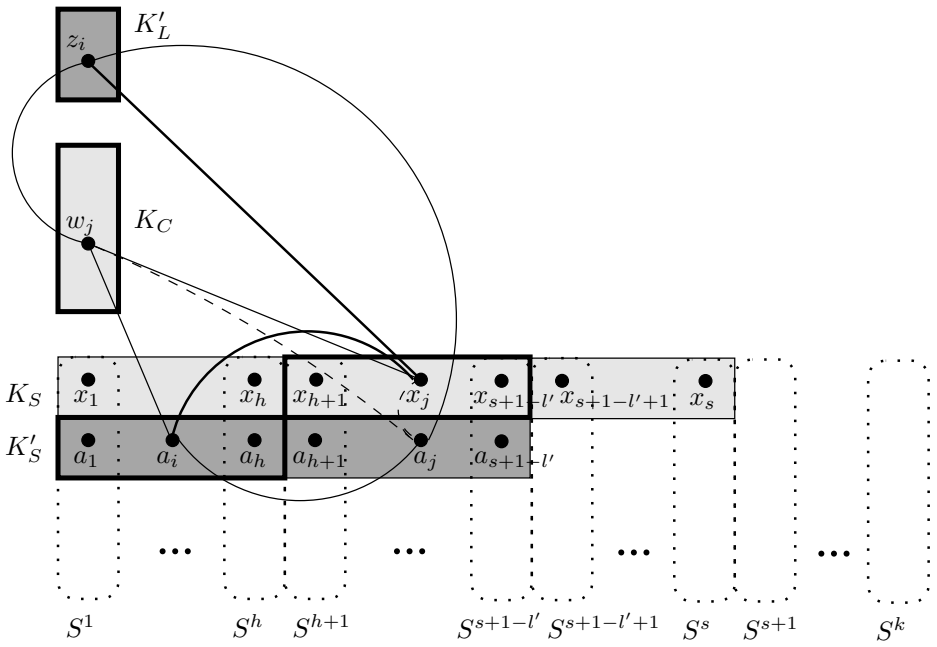


Figure 3.1: Proof of Theorem 3.2, implication *1.* $\Rightarrow$ *3.*

Fact 3.4. *$K_C \cup K'_L \cup \{x_1, \ldots, x_{s+1-l'}\}$ has $|K_C| + s + 1$ vertices thus it cannot be a clique $\Rightarrow h \geq 1$.*

Fact 3.5. *$\forall i \leq h,\ \forall j \geq h+1$, $\{a_j, a_i, w_j, x_j\}$ does not induce a $P_4 \Rightarrow\ a_i x_j \in E$.*

The above facts are illustrated in Figure 3.1 where dashed lines are forbidden edges and thick lines are the necessary edges. Then, it suffices to remark the following: Fact 3.3 and Fact 3.5 imply that $K_C \cup K'_L \cup \{a_j, j \leq h\} \cup \{x_j, h+1 \leq j \leq s+1-l'\}$ is a clique of cardinality $|K|+1$, represented by the sets with bold frames, which yields a contradiction. □

The above result shows in particular that *Statement 3* is true neither for comparability graphs (known also as transitively orientable graphs) which contain cographs as a subclass, nor for triangulated graphs.

Note also that, by the auto-complementarity of cographs, we have trivially that for all $V' \subseteq V$ such that $G[V'] \in \mathcal{K}_p\mathcal{S}_k$ where $k \geq 1$, then $G[V' \backslash S] \in \mathcal{K}_p\mathcal{S}_{k-1}$ where S is a maximum stable set of $G[V']$.

One can observe in Figure 3.2 that in Theorem 3.2, if we take a maximal stable set instead of a maximum one, then the statement does not hold. The graph G of Figure 3.2 is clearly $(1,1)$-colorable. But, $G[V \setminus S]$, where S is a maximal stable set of G, is not $(1,0)$-colorable.

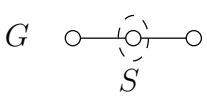


Figure 3.2: Theorem 3.2 does not hold with a maximal stable set.

The proof of the following corollaries of Theorem 3.2, suggests some algorithms to solve problems related to (p,k)-coloring.

Corollary 3.6. *In any cograph G:*

1. *for any k, $p_{min} = \min\{p : G \in \mathcal{K}_p\mathcal{S}_k\}$ can be found in time $\mathcal{O}(n^2 + nm)$;*
2. *Min Split-coloring can be solved in time $\mathcal{O}(n^2 + nm)$.*

Proof. For *Statement 1*, the algorithm consists in repetitively constructing a maximum clique in the remaining graph and removing it as long as it is of size at least $k+1$. Then, $S_1, \ldots, S_k$ are determined by a greedy coloring algorithm on the remaining graph. It suffices to observe that according to Theorem 3.2, when the algorithm has already set p_{min} cliques, the remaining graph is in $\mathcal{K}_0\mathcal{S}_k$ and thus does not admit a clique of size $k+1$. Note that the only computation consists in finding p_{min} maximum cliques, which takes $\mathcal{O}(p_{min}(n+m))$ time, and running once the greedy coloring algorithm in the remaining graph, which takes only linear time.
As for *Statement 2*, in step i of the algorithm, one will compute a maximum clique in the remaining graph; if it is of size at least $i+1$ then it will be chosen and removed from the graph before going to step $i+1$, otherwise an i-coloring will be computed in the remaining graph. To conclude, note that in step i, the number of fixed cliques is i. Therefore, if $\chi_S(G) = k$ then in step k, the remaining graph belongs to $\mathcal{K}_0\mathcal{S}_k$. Obviously, the time complexity is the same as the previous part of the corollary. □

It is needless to say that (p, k_{min})-coloring is solved in a similar way with the same time complexity in cographs.
One can also generalize the above corollary in the following way.

Corollary 3.7. *For any cograph G and any function $f : \mathbb{N} \longrightarrow \mathbb{N}$, $p_0 = Min\{p : G \in \mathcal{K}_p\mathcal{S}_{f(p)}\}$ can be computed in polynomial time.*

Proof. First of all, the existence of a p such that $G \in \mathcal{K}_p\mathcal{S}_{f(p)}$ is obvious since $\mathcal{K}_p\mathcal{S}_i \subset \mathcal{K}_p\mathcal{S}_{i+1}$. Now, let us consider the same algorithm as in *Statement 2* of Corollary 3.6 with the only difference that we test if the maximum clique in step i is of size at least $f(i)+1$ (and not $i+1$), and consequently the algorithm ends with an optimal coloring of the rest of the graph in $f(i)$ (and not i) colors. Once again, at step i we have set exactly i cliques. To complete the proof, it suffices to remark that after p_{min} steps ($i = p_{min}$), the remaining graph belongs to $\mathcal{K}_0\mathcal{S}_{f(p)}$.
Note that for a constant function f, we obtain *Statement 1* of Corollary 3.6; $f(p) = p$ gives the second part; $f = 0$ boils down to the problem of vertex covering by cliques and finally if $f = \chi(G)$ then $p_{min} = 0$, i.e., $G \in \mathcal{K}_0\mathcal{S}_{\chi(G)}$.
Finally, this result is also true when interchanging cliques and stables sets by simply applying the same algorithm to the complement of G. □

3.4 Cocoloring of cographs

Min Cocoloring has been solved in time $\mathcal{O}(n^2)$ for cographs with a dynamic programming procedure visiting the vertices of their associated cotree from the leaves to the root [78]. In what follows, we use Theorem 3.2 to devise a simpler algorithm for Min Cocoloring in the class of cographs. More precisely, we show that the greedy algorithm for cocoloring given in [65] is exact for cographs and has a better time complexity than $\mathcal{O}(n^2)$. It runs as follows.

Algorithm 8 Greedy cocoloring

Input: a graph $G = (V, E)$

Output: a cocoloring $\mathcal{Z}$ of G

1. $\mathcal{Z} \leftarrow \emptyset$;
2. **while** $G \neq \emptyset$ **do**
3. compute a maximum stable set S and a maximum clique K of G;
4. choose $X = \text{argmax}(|S|, |K|)$ and $\mathcal{Z} \leftarrow \mathcal{Z} \cup \{X\}$;
5. $G \leftarrow G[V \setminus X]$.

Theorem 3.8. *For any cograph G given by its cotree, a minimum cocoloring is obtained by the greedy cocoloring algorithm in time $\mathcal{O}(n^{3/2})$.*

Proof. Let p, k be such that $G \in \mathcal{K}_p\mathcal{S}_k$ and that $z(G) = p + k$. Assume that the greedy algorithm does not give an optimal cocoloring. This means that there is a first step where we cannot get an optimal decomposition anymore. At this step, assume that we have already removed p' cliques and k' stable sets, and that we remove a maximum clique K without loss of generality. If G' is the remaining graph after $p' + k'$ removals, Theorem 3.2 implies that $G' \in \mathcal{K}_{p-p'}\mathcal{S}_{k-k'}$. By hypothesis, we have $z(G') = z(G) - p' - k'$ but $z(G' \setminus K) > z(G) - p' - k' - 1$. Note that we cannot have $p - p' < 0$ since at some earlier stage we would have faced the problem, neither $p - p' > 0$ because the problem would not occur yet. Therefore, we must necessarily have $p - p' = 0$. Now, one can derive the following inequalities using the facts that $G' \in \mathcal{K}_0\mathcal{S}_{k-k'}$ and $z(G') = \chi(G') = \omega(G') = k - k'$:

$$\alpha(G') - 1 = \theta(G') - 1 \geq z(G' \setminus K) > z(G') - 1 = \chi(G') - 1 = \omega(G') - 1.$$

Hence, the greedy algorithm should have chosen S instead of K which means that this case is not possible.

As for the time complexity, it is well known that a maximum clique and a maximum stable set can be found on a cotree in time $\mathcal{O}(n)$. On the other hand, the number of steps is exactly $z(G)$. If we denote by n_i the number of vertices removed at step i then the inequality $z(G) - (i - 1) \leq n_i$ holds for $i = 1, \ldots, z(G)$. Summing both sides of these inequalities for every i yields the bound $\mathcal{O}(\sqrt{n})$ on $z(G)$, which gives $\mathcal{O}(n^{3/2})$ as overall complexity. □

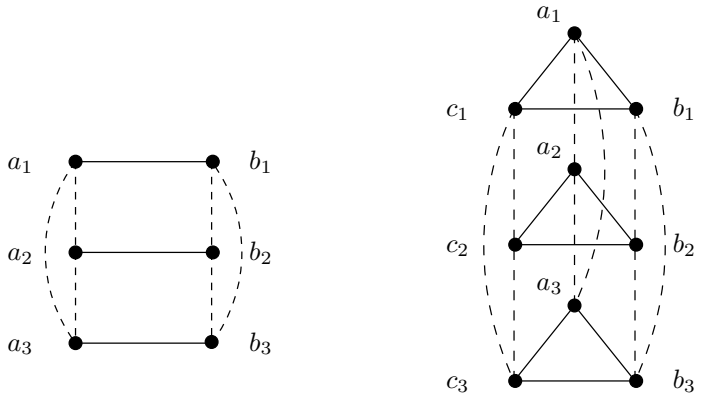
3.5 Characterizing $(2,1)$- and $(2,2)$-colorable cographs

In this section, we give forbidden configurations characterizing $(2,1)$- and $(2,2)$-colorable cographs. In this purpose, let us point out the following lemma.

Lemma 3.9. *Every connected component of a triangle-free cograph is a complete bipartite graph.*

Proof. Let $G = (A \cup B, E)$ denote a triangle-free cograph which is obviously a bipartite graph. Consider a connected component of G where one vertex $a_1 \in A$ is adjacent to b_1 and to b_2 of B. It is easy to verify that if one of the neighbors of a_1 is adjacent to a vertex $a_2 \in A$ then any other neighbor of a_1 has to be adjacent to a_2 as well, in order to avoid any possible P_4. Applying this argument to the vertices of both A and B yields a complete bipartite graph. □

In what follows, we denote by H the bipartite graph $H = (A \cup B, E_H)$ where $A = \{a_1, a_2, a_3\}$, $B = \{b_1, b_2, b_3\}$ and $E_H = \{a_1b_1, a_2b_2, a_3b_3\}$. L is the tripartite graph $L = (A \cup B \cup C, E_L)$ where $A = \{a_1, a_2, a_3\}$, $B = \{b_1, b_2, b_3\}$, $C = \{c_1, c_2, c_3\}$ and $E_L = \{a_ib_i, a_ic_i, b_ic_i, i = 1, 2, 3\}$. Both of these configurations are shown in Figure 3.3 where dashed lines represent forbidden edges and edges which do not appear in the figures are optional edges. One can notice that as soon as one of the optional edges links two connected components of H or L, then, by Lemma 3.9, all of the optional edges between these connected components are present.



a) H b) L

Figure 3.3: Configurations H and L.

Proposition 3.10. *A cograph is (2,1)-colorable if and only if it does not contain configuration H.*

Proof. If G contains $H = (V_H, E_H)$ as configuration, then the subgraph $G[V_H]$ is clearly not $(2,1)$-colorable. Conversely, let us suppose that G does not contain configuration H. Let

S^* be a maximum stable set. Then, let S' be a maximum stable set of $G[V \setminus S^*]$, $G[S' \cup S^*]$ is bipartite and admits S^* as maximum stable set; hence S' is a minimum vertex cover (a set of vertices covering all the edges of the graph) and there exists a matching of size $|S'|$ by König's theorem[3]. Since G does not contain H, this matching is of size at most 2. It means that $\alpha(G[V \setminus S^*]) \leq 2$ and consequently (since cographs are perfect), $G[V \setminus S^*] \in \mathcal{K}_2\mathcal{S}_0$, which implies $G \in \mathcal{K}_2\mathcal{S}_1$. □

Proposition 3.11. *A cograph is (2,2)-colorable if and only if it does not contain configuration L.*

Proof. If G contains $L = (V_L, E_L)$ as configuration, then $G[V_L]$ is clearly not $(2,2)$-colorable. Conversely, let us suppose that $G \notin \mathcal{K}_2\mathcal{S}_2$. Let S^* be a maximum stable set of G, according to Theorem 3.2, $G[V \setminus S^*] \notin \mathcal{K}_2\mathcal{S}_1$. Consequently (Proposition 3.10), $\exists S^1 \cup S^2 \subset V \setminus S^*$, such that S^1 and S^2 are stable, $|S^1| = |S^2| = 3$ and $G[S^1 \cup S^2]$ admits a perfect matching. Since S^* is also a maximum stable set of the cograph $G' = G[S^* \cup S^1 \cup S^2]$, this graph is not $(2,2)$-colorable. In particular, it is not bipartite, consequently it contains a triangle (a', b', c') with $a' \in S^*$, $b' \in S^1$ and $c' \in S^2$. (a', b', c') is a maximum clique of G', consequently (Theorem 3.2) $G'' = G[(S^* \cup S^1 \cup S^2) \setminus \{a', b', c'\}] \notin \mathcal{K}_1\mathcal{S}_2$. Thus, G'' contains also a triangle (a'', b'', c'') with $a'' \in S^*$, $b'' \in S^1$ and $c'' \in S^2$, and $G''' = G[(S^* \cup S^1 \cup S^2) \setminus \{a', a'', b', b'', c', c''\}] \notin \mathcal{K}_0\mathcal{S}_2$ and contains a triangle (a''', b''', c''') with $a''' \in S^*$, $b''' \in S^1$ and $c''' \in S^2$. The subgraph $G[\{a', a'', a''', b', b'', b''', c', c'', c'''\}]$ contains configuration L, which concludes the proof. □

3.6 Characterizing (p, k)-colorable cographs

Following the above ideas, the forbidden configurations given in the previous section can be generalized for (p,k)-colorable cographs for any given p and k (see Francisco et al. [66]). In what follows, let $l * K_r$ denote the configuration formed by l copies of a clique of size r; if K^i is a clique of size r having vertices v^i_j, where $j = 1, \ldots, r$ then the sets $\{v^1_j, \ldots, v^l_j\}$, where $j = 1, \ldots, r$, are stable sets and the remaining edges are optional. For instance, the configurations H and L are respectively $3 * K_2$ and $3 * K_3$.
Francisco et al. first restate Lemma 3.9 with forbidden cliques of size l.

Lemma 3.12 ([66]). *Every connected component of a cograph without K_l is a $(l-1)$-partite graph, for $l > 2$.*

Their second lemma consists in a generalization of Proposition 3.10.

Lemma 3.13 ([66]). *A cograph G is $(p,1)$-colorable if and only if it does not contain the configuration $(p+1) * K_2$.*

[3]For a bipartite graph G, the maximum number of edges in a matching is equal to the minimum number of vertices in a vertex cover.

Finally, they come up with the following theorem.

Theorem 3.14 ([66, 57]). *A cograph G is (p,k)-colorable if and only if it does not contain $(p+1) * K_{k+1}$ as an induced subgraph.* □

3.7 Concluding remarks

The results obtained in this chapter show that generalized coloring problems are rather well solved in the class of cographs; they admit simple polynomial time algorithms. In particular, we show that the greedy cocoloring algorithm (given in Algorithm 8) provides an optimal cocoloring of a graph G if and only if G is a cograph.

Other results on partitioning the vertex set of a cograph into cliques and stable sets concern the more general framework of matrix partitions studied in [57] by Feder et al. Let M be an m by m matrix over $0, 1, *$. An *M-partition* of a graph G is a partition of its vertex set into m parts $V_1, V_2, \ldots, V_m$ such that each V_i is a clique (respectively a stable set) whenever $M(i,i) = 1$ (respectively $M(i,i) = 0$). There are all possible edges (respectively no edge) between parts V_i and V_j whenever $M(i,j) = 1$ (respectively $M(i,j) = 0$), and no condition is required on the edges between V_i and V_j if $M(i,j) = *$. When a graph does not admit an M-partition, it is said to be an *M-obstruction*. The authors also consider the M-partitioning problem with lists, i.e., under the condition that each vertex can be assigned to a restricted list of possible parts. A *constant matrix* is called as *(a,b,c)-block matrix* if a, b and c are chosen from $0, 1$ and $*$, and there is no diagonal *'s; a indicates the links between stable sets, b between cliques and c between stable sets and cliques. Then, it is clear that a graph is (p,k)-colorable if and only if it admits an M-partition, where M is a $(*,*,*)$-block matrix with p cliques and k stable sets. In [57], the authors give upper bounds (depending on p and k) on the size of a largest minimal M-obstruction for different types of matrices M. They obtain as a result that in cographs, any M-partition problem can be solved in polynomial time for any fixed number of cliques and stable sets. In the case where M is a $(*,*,*)$-block matrix, they precisely show that any minimal obstruction has exactly $(p+1)(k+1)$ vertices and, furthermore, it is both $(k+1)$-colorable and partitionable into $(p+1)$ cliques; describing the forbidden configurations given in Theorem 3.14.[4]

One can remark that polar graphs can also be expressed in terms of matrix partitions; a graph is (s,t)-polar if and only if it admits an M-partition where M is a $(1,0,*)$-block matrix with s stable sets and t cliques. In the next chapter, we explore polar cographs in terms of forbidden subgraphs and recognition algorithms.

Note finally that all M-partition problems are expressible in monadic second-order logic, and hence solvable in polynomial time for bounded treewidth graphs (see[32]). It is also shown in [33] that bounded cliquewidth graphs (and in particular cographs) allow a polynomial time solution for finding maximum subgraphs with any monadic second-order logic property (e.g. $\alpha_{p,k}$ for fixed p and k, or a maximum (s,t)-polar subgraph for fixed s and t).

[4] Note that list matrix partitions of triangulated graphs (see [59]) and perfect graphs (see [56]) are also studied in the same spirit.

Chapter 4

Polar cographs

As mentioned at the end of the previous chapter, recognizing polar graphs can be seen as a generalized graph coloring problem. Recognizing polar graphs amounts indeed to determining whether a given graph admits a partition of its vertices into stable sets and cliques allowing only some particular links among stable sets and among cliques; namely, stable sets have to be completely linked to each other, cliques not linked at all to each other, and no constraint is required on the links between cliques and stable sets.
In [21] it is shown that recognizing whether an arbitrary graph is polar is $\mathcal{NP}$-complete. Thus, these graphs have not been extensively studied and no good characterization is known. Previous works on polar graphs concern some polynomial time recognition problems. In [103], the authors give a polynomial time recognition algorithm for polar graphs where the largest sizes of the stable sets and of the cliques are bounded. In this case, a result of [124] suggests that there is a finite list of forbidden subgraphs characterizing such polar graphs. Also, in [114], the authors claim that there is a polynomial time algorithm to recognize polar graphs, which admit a polar partition where all the cliques are of size 1. According to [69], polar graphs with stable sets of size 1 and cliques of size at most 2 admit a characterization with 18 forbidden subgraphs.
Alekseev et al. [3] set the complexity status of $(\mathcal{P}_1, \ldots, \mathcal{P}_p, \mathcal{Q}_1, \ldots, \mathcal{Q}_k)$-coloring problems[1] for different types of properties. The authors point out that there is an unexplored gap when $\mathcal{P}_i$'s are additive hereditary, and $\mathcal{Q}_j$'s are co-additive hereditary[2]. We notice that the recognition of polar graphs is exactly in this gap since it corresponds to (P_3-free, $\overline{P_3}$-free)-coloring.
The question which arises is to find restricted classes of graphs where polar graphs can be recognized in polynomial time and admit nice characterizations. It turns out that for cographs, one can derive results of such type; this is a first step which could be followed by various extensions.
In this chapter, based on [51], we handle polar cographs and we provide their characterization

[1] Recall that a graph is $(\mathcal{P}_1, \ldots, \mathcal{P}_p, \mathcal{Q}_1, \ldots, \mathcal{Q}_k)$-colorable if it admits a partition $(U_1, \ldots, U_p, W_1, \ldots, W_k)$ of its vertex set, where each U_i verifies the property $\mathcal{P}_i$ and each W_j verifies the property $\mathcal{Q}_j$.
[2] See Subsection 1.2.2 for the definitions of additive and co-additive properties.

in terms of forbidden subgraphs in Section 4.2. We also examine the monopolar cographs which are the (s,t)-polar cographs where $\min(s,t) \leq 1$. A characterization of these graphs by forbidden subgraphs is given in Section 4.3. Besides, in Section 4.4, we give a polynomial time algorithm for finding a largest induced polar subgraph in cographs using dynamic programming on cotrees. Finally, we derive a recognition algorithm for polar cographs in time $\mathcal{O}(n \log n)$ in Section 4.5.

4.1 Introduction

Before discussing briefly some properties of polar graphs, let us first recall their definition.

Definition 4.1 (polar graph, polar partition). *A graph $G = (V, E)$ is called polar if its vertex set V can be partitioned into (A, B) (A or B may possibly be empty) such that A induces a complete multipartite graph (it is a join of stable sets) and B a (disjoint) union of cliques (i.e., the complement of a join of stable sets). Such a partition (A, B) is called a polar partition.*

According to [21], a polar graph with polar partition (A, B) is said to belong to the class (α, β) if the size of a stable set in A does not exceed α and the size of a clique in B does not exceed β. Here, we classify polar graphs rather in terms of numbers of stable sets and cliques in a polar partition (A, B) that they admit. This choice is justified by both the results obtained in this chapter and the sake of compatibility with the notion of (p, k)-colorability used in the rest of the dissertation. However, although inverted with respect to our basic definition of (p, k)-colorability, in this chapter, we adopt a notation indicating first stable sets and then cliques as it was used in [21].

Definition 4.2 ((s,t)-polar). *A graph is (s,t)-polar if it admits a polar partition (A, B) where A is a join of at most s stable sets and B a union of at most t cliques.*

Thus polar graphs are just the (∞, ∞)-polar graphs meaning that the numbers of stable sets and cliques in a polar partition are unbounded. Observe that the complement $\overline{G}$ of an (s,t)-polar graph is a (t,s)-polar graph.
Clearly, polar graphs are closely related to (p, k)-colorable graphs; if a graph is (s,t)-polar then it is obviously (t,s)-colorable. But the converse does not necessarily hold since an (s,t)-polar graph is a graph admitting only a particular (t,s)-coloring. Moreover, not every graph is polar: the graphs N_1 and N_2 in Figure 4.1 are not polar as can be checked, but if any vertex is removed, the remaining graph is polar.

We will also examine a subclass of polar graphs defined as follows.

Definition 4.3 (monopolar graphs). *A graph is monopolar if it is (s,t)-polar for some s and t such that* $\min(s,t) \leq 1$.

In other words, for a monopolar graph, a partition (A, B) exists with at most one stable set in A or at most one clique in B.

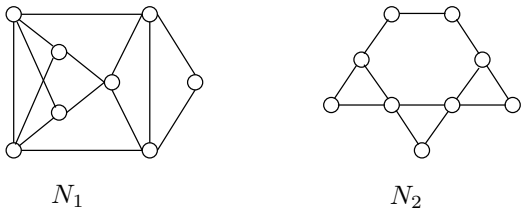


Figure 4.1: Some minimal non-polar graphs.

It will be convenient to denote by $K^*(s,t)$ an (s,t)-polar graph with partition (A,B), which is the join of A and B (i.e., with complete links between A and B). This graph is called a *complete* (s,t)-polar graph.
Basic properties of cographs are already defined in the previous chapter; here we use the same terminology. Let us mention some additional notations that we frequently use in this chapter. Given two graphs G_1 and G_2, $G_1 \oplus G_2$ denotes their join (with complete links) and $G_1 \cup G_2$ their disjoint union (as already introduced). Let x and y be two vertices, in this chapter, for the sake of simplicity, xy and $\overline{xy}$ respectively mean that they are adjacent and non-adjacent.

4.2 Characterization of polar cographs by forbidden subgraphs

In this section, we provide a forbidden subgraph characterization of polar cographs. Since there is a finite family of forbidden subgraphs, there is an obvious polynomial time recognition algorithm. We will however describe in Section 4.5 a recognition algorithm with better time complexity.

Theorem 4.4. *For a cograph G, the following statements are equivalent:*

1. *G is polar;*
2. *neither G nor $\overline{G}$ contains any one of the graphs $H_1, \ldots, H_4$ of Figure 4.2 as induced subgraphs.*

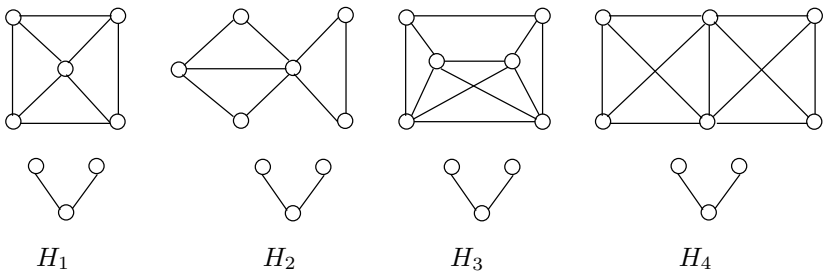


Figure 4.2: Forbidden subgraphs for polar cographs.

Proof. *1.* $\Rightarrow$ *2.* Since every induced subgraph of a polar graph is polar and the complement of a polar graph is polar, it is enough to show that $H_1, \ldots, H_4$ are non-polar. Suppose $H_1, \ldots, H_4$ are polar. First, it is routine to check that none of $H_1, \ldots, H_4$ admits a polar partition with one stable set. Furthermore, since a complete join of stable sets is a connected subgraph, it follows that in each of the four graphs, one of the components must induce a disjoint union of cliques. Clearly, it is not the case for any of the graphs. Hence they are non-polar.
2. $\Rightarrow$ *1.* Suppose G is non-polar. Assume without loss of generality that G is minimal non-polar. Assume also without loss of generality that the cograph G is disconnected (otherwise take its complement). Let (A, B) be a partition of its vertex set into non-empty sets without edges between A and B. By the minimality of G, both $G[A]$ and $G[B]$ are polar. If $G[A]$ contains no induced P_3, then it is a disjoint union of cliques and hence G is polar. Hence, we may assume that both A and B contain three vertices inducing a P_3. If both $G[A]$ and $G[B]$ have polar partitions with single stable sets S_A and S_B respectively, then G has a polar partition with single stable set $S_A \cup S_B$. Thus, assume that $G[A]$ has at least two stable sets in every polar partition. Let $A' \subset A$ induce the connected component containing the join of stable sets (in a possible polar partition of A). Then, $A \setminus A'$ induces a disjoint union of cliques. Since G is a cograph, A' is partitioned into (C, D) with complete join between C and D. We consider two cases.

Case 1. An induced P_3 in A is completely contained in D.

If C contains a non-edge, then the non-edge along with the P_3 in D and the P_3 in B induce an H_1 in G, a contradiction. Thus, C must induce a clique. If C contains an edge, then D is $2K_2$-free, for otherwise G contains an H_4. D is also C_4-free (else G contains H_1) and P_4-free. Thus D induces a threshold graph. It follows that $G[A]$ has a polar partition with at most one stable set and many cliques, which is a contradiction. It follows that C must consist of a single vertex.
Let S be a maximal stable set in D containing both the ends of P_3 in D. Let c be the center of the P_3.

Claim 4.5. *For any $a \in D \setminus S$, ac, i.e., $D \setminus S \subseteq N(c)$.*

Proof. S being a maximal stable set, a has a neighbor in S. If $a \notin N(c)$, then D must contain a P_4, C_4 or $P_3 \cup K_2$, which is a contradiction since G is P_4-free, H_1-free and H_2-free. □

Claim 4.6. *For any $a \in D \setminus S$, ax, for some $x \in N(c) \cap S$.*

Proof. Similar to the proof of Claim 4.5. □

Claim 4.7. *$N(c) \cap S$ is linearly ordered by domination in $N(c) \setminus S$, i.e., there are no two vertices $x, y \in N(c) \cap S$ such that for some $a, b \in N(c) \setminus S$, $xa, \overline{xb}, \overline{ya}$ and yb.*

Proof. Since $a, b \in N(c) \setminus S$, ac and bc. If $\overline{ab}$ then a, b, c, x, y and the vertex in C along with P_3 in B induce H_4. If ab, then G contains a P_4. □

Claim 4.8. *There exists $d \in N(c) \cap S$ such that da for all $a \in N(c) \setminus S$.*

Proof. Follows from Claim 4.6 and 4.7. □

Claim 4.9. *For $x \in S \setminus \{d\}$ and for $a, b \in N(c) \setminus S$, if $\overline{ab}$ then $\overline{xa}$ and $\overline{xb}$.*

Proof. Since da and db, it follows that if xa and/or xb, then D induces a C_4 or P_4, which is a contradiction. □

Claim 4.10. *$N(c) \setminus S$ is $2K_2$-free.*

Proof. Any $2K_2$ in $N(c) \setminus S$, along with cd, the vertex in C and a P_3 from B would induce H_4 in G, which is a contradiction. □

Claim 4.11. *$G[A]$ has a polar partition with a single stable set.*

Proof. By Claim 4.10, $N(c) \setminus S$ is $2K_2$-free. Also D is C_4-free. Hence $N(c) \setminus S$ induces a threshold graph. Let (S', K) be a polar partition of $(N(c) \setminus S) \cup C$ with S' the single stable set and K the single clique. Then $(S' \cup S \setminus \{d\}, K \cup \{d\})$ is a polar partition of $G[A]$ with a single stable set, by Claims 4.8 and 4.9. □

It follows that Case 1 is impossible.

Case 2. Every P_3 of A intersects both C and D.

Since both C and D are P_3-free, each one induces a disjoint union of cliques. We can assume without loss of generality that C consists of either a single clique or a single stable set, for otherwise, i.e., if both C and D are neither a single clique nor a single stable set, G contains H_3. If C consists of a single stable set, then $G[A]$ has a polar partition with one stable set. If C consists of a clique of size at least 2, then D has at most one clique of size at most 2 (else G has H_4). It follows that the rest of D forms a single stable set and $G[A]$ has a polar partition with a single stable set and many cliques. Thus this case is also impossible.

It follows that G must be polar. □

4.3 Characterization of monopolar cographs by forbidden subgraphs

As in Section 4.2, we give here a characterization based on forbidden subgraphs; a more involved recognition algorithm for monopolar graphs will be given in Section 4.5.

Theorem 4.12. *For a cograph G, the following statements are equivalent:*

1. *G is monopolar;*
2. *neither G nor $\overline{G}$ contains any one of the graphs $G_1, \ldots, G_9$ of Figure 4.3 as an induced subgraph;*
3. *G or $\overline{G}$ is a disjoint union of threshold graphs and complete $(1, \infty)$-polar graphs.*

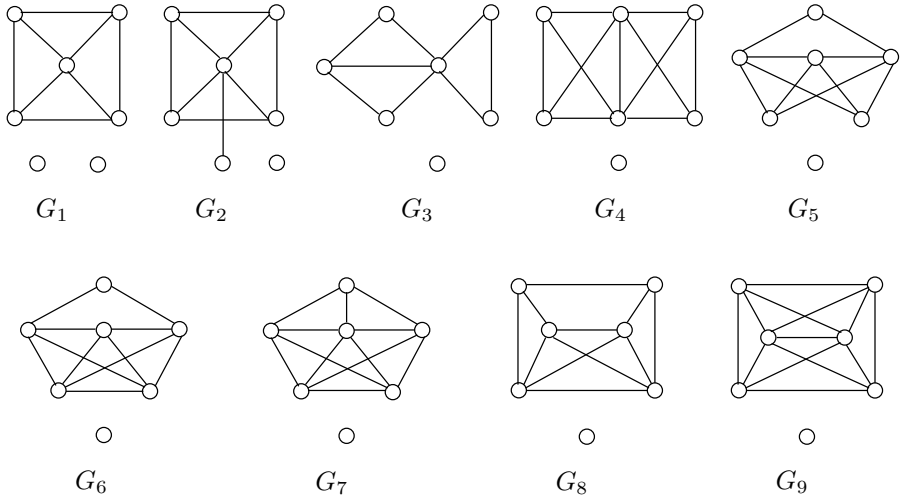


Figure 4.3: Forbidden subgraphs for monopolar cographs.

Proof. 1. $\Rightarrow$ 2. Since the complement of a monopolar graph and every induced subgraph of a monopolar graph are also monopolar, it is enough to show that $G_1, \ldots, G_9$ are not monopolar. Since the non-trivial component in each of these graphs is not a disjoint union of cliques, it must contain the join of stable sets in any polar partition. It is routine to verify that any polar partition of these graphs must be the join of at least two stable sets and the union of at least two cliques. Hence they are not monopolar.
3. $\Rightarrow$ 1. Since a threshold graph has a polar partition into a single stable set and a single clique, and since disjoint union of stable sets is a single stable set, it follows that if G is a disjoint union of threshold graphs and complete $(1, \infty)$-polar graphs, then G is monopolar with a single stable set and a disjoint union of cliques in a polar partition.
2. $\Rightarrow$ 3. Since G is a cograph, assume without loss of generality that G is disconnected. It is enough to show that each non-trivial component of G is either a threshold graph or a complete $(1, \infty)$-polar graph. Let G' be any non-trivial component of G. Further assume that G' is a join of A and B (i.e., $G' = A \oplus B$). The non-empty graphs A, B exist since G' is a connected cograph with at least two vertices. We consider several cases.

Case 1. A contains an induced C_4 with vertices a, b, c, d and edges ab, bc, cd and ad.
B must be a stable set, for otherwise G contains $G_9 = (C_4 \oplus K_2) \cup K_1$. Let x be any other vertex of A. Then,

i) x must be joined to at least one vertex of the C_4, for otherwise G contains $G_2 = ((C_4 \cup K_1) \oplus K_1) \cup K_1$;

ii) x may not be joined to exactly three vertices of the C_4, for otherwise G contains G_7;

iii) x may not be joined to all four vertices of the C_4, for otherwise G contains $G_9 = ((C_4 \oplus K_1) \oplus K_1) \cup K_1$;

iv) x may not miss two adjacent vertices of the C_4, for otherwise G' contains P_4.

It follows that each vertex of A other than a, b, c, d is joined either to a and c, or else is joined to b and d. Let N_a be the set of all neighbors of a in A and N_b be the set of all neighbors of b in A. Clearly N_a and N_b form stable sets by i)-iv) and are also completely joined, to avoid induced P_4. Thus G' is a join of three stable sets. Now G may contain at most one other component which must be a clique, for otherwise G contains G_1. It follows that the complement $\overline{G}$ is a complete $(1,3)$-polar graph in this case, as required.

Case 2. A contains an induced $2K_2$.

B must form a stable set, for otherwise G contains $G_4 = (2K_2 \oplus K_2) \cup K_1$. If A contains an induced P_3, then to avoid P_4, it must contain P or Q of Figure 4.4 as an induced subgraph. If A contains P, then G contains $G_4 = (P \oplus K_1) \cup K_1$, and if A contains Q, then G contains $G_3 = (Q \oplus K_1) \cup K_1$. It follows that A is P_3-free and hence induces a disjoint union of cliques. Hence $G' = A \oplus B$ is a complete $(1, \infty)$-polar graph as required.

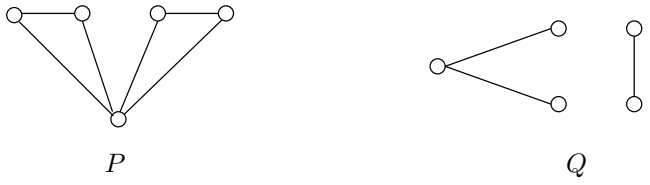


Figure 4.4: Case 2 of Theorem 4.12.

We may now assume by symmetry that both A and B do not contain induced $C_4, 2K_2$ and P_4, and hence form threshold graphs.

Case 3. A is a threshold graph containing a triangle.

i) If A contains a $K_4 \setminus e$, i.e., a K_4 where an edge is removed, then B must be a clique, or else G contains $G_9 = ((K_4 \setminus e) \oplus 2K_2) \cup K_1$. Since a threshold graph joined to a clique is a threshold graph, G' is a threshold graph in this case, as required.

ii) If A contains a vertex joined to exactly one vertex of the triangle, then B must be a clique too, or else G contains G_7. Hence G' is a threshold graph, as required. It follows that A induces a clique and isolated vertices.

iii) If A forms a clique and at least one isolated vertex, then B contains no non-edge, or else G contains G_6. Thus B is a clique and hence G' is a threshold graph, as required.

iv) If A forms a single clique, then B being a threshold graph, G' too is a threshold graph, as required.

Case 4. Both A and B induce threshold graphs with no triangles.

i) If A contains an induced $P_3 \cup K_1$, then B must be a clique, for otherwise G contains $G_5 = ((P_3 \cup K_1) \oplus 2K_1) \cup K_1$. Hence G' is a threshold graph, as required. So we may assume that A is either $K_{1,n}$ for some $n > 1$, or P_3-free.

ii) If A is $K_{1,n}$ with $n > 1$, then B may not contain an induced P_3 (to avoid $G_9 = (P_3 \oplus P_3) \cup K_1$) and may not contain an induced $K_2 \cup K_1$ (to avoid $G_7 = (P_3 \oplus (K_2 \cup K_1)) \cup K_1$). Thus B is a single clique or a single stable set. If B is a clique, then G' is a threshold graph, as required, and if B is a stable set with at least two vertices, then G may contain only one other component which is a clique, or else G contains G_1. Hence $\overline{G}$ is a complete $(1,3)$-polar graph, as required.

iii) Therefore, by symmetry, we may assume that both A and B may not contain $K_2 \cup K_1$, for otherwise G contains $G_8 = ((K_2 \cup K_1) \oplus (K_2 \cup K_1)) \cup K_1$. Thus, one of A and B, say B, is a clique or a stable set. If B is a clique then, since A may not contain $2K_2$ (otherwise G contains $G_4 = (2K_2 \oplus K_2) \cup K_1$), G' is a threshold graph as required. If B is a stable set, then G' is a complete $(1,\infty)$-polar graph since A is a disjoint union of cliques.

Thus in all cases, either the complement $\overline{G}$ is a complete $(1,3)$-polar graph or G is a disjoint union of threshold graphs and complete $(1,\infty)$-polar graphs. □

4.4 Largest polar subgraph in cographs

In this section, we describe how to find an induced polar subgraph of maximum size in a cograph using its cotree representation. Given a cograph G, let us denote by $MC(G)$ a maximum clique in G, by $MS(G)$ a maximum stable set in G, by $MT(G)$ a maximum threshold graph in G, by $MUC(G)$ a maximum (size) union of cliques in G, by $MJS(G)$ a maximum (size) join of stable sets in G, by $MMPS(G)$ a maximum $(1,t)$-polar subgraph for some t (maximum monopolar subgraph with one stable set) in G, by $MMPC(G)$ a maximum $(s,1)$-polar subgraph for some s (maximum monopolar subgraph with one clique) in G, and finally by $MP(G)$ a maximum polar subgraph in G. $n(MP(G))$ denotes the size of $MP(G)$ and the sizes of all other maximum subgraphs are denoted in a similar way. All maximum subgraphs mentioned below are represented by a pair (A,B) as described in the introduction.

In what follows, we assume that the cotree representation of the cograph is given. As

previously, each vertex x in the cotree has a type $t(x)$ which is 0 or 1, and $c_1x, c_2x, \ldots$ are the children of x. For simplicity purposes, a vertex x will also represent the subgraph associated with vertex x of the cotree; this should be clear from the context.
First of all, note that given a cotree, $MS(x)$ and $MC(x)$ can be found in linear time for any x in the cotree [29]. Also, it has been shown in Theorem 3.2 that a maximum threshold subgraph in cographs is obtained by the union of any maximum stable set and any maximum clique, since no pair of maximum stable set and maximum clique is disjoint. Therefore, for any vertex x of the cotree, $MT(x)$ is the subgraph induced by the vertices of $MC(x) \cup MS(x)$ and $n(MT(x)) = n(MC(x)) + n(MS(x)) - 1$. In what follows, we assume for the sake of simplicity that $MC(x), MS(x)$ and $MT(x)$ are known for any vertex x of the cotree.
Note that the following lemmas are established for computing a maximum subgraph at a 0-vertex, assuming that all parameters on children needed for the computation are already known. Consequently, the maximum subgraph M computed at a 0-vertex has a polar partition (A, B) such that: if M is the result of a union of maximum subgraphs M_i with polar partition (A_i, B_i), where A_i induces one stable set in child c_ix, then $A = \cup_i A_i$ (inducing one stable set) and $B = \cup_i B_i$ (inducing the union of cliques of each M_i); if M is a subgraph M_j with polar partition (A_j, B_j) realizing the maximum of some quantity, then $A = A_j$ and $B = B_j$ (knowing that a sum in a maximum operator should be interpreted as a union of the considered subgraphs).

Lemma 4.13. *Given a cotree, $MUC(x)$ and $MJS(x)$ can be computed for any 0-vertex x in time linear in the number of children of x.*

Proof. Clearly, we have $MUC(x) = \cup_i MUC(c_ix)$ since cliques of different children are not linked at all. On the other hand, $MJS(x)$ is the set realizing the maximum of

$$\big[\max_i n(MJS(c_ix)), \sum_i n(MS(c_ix))\big].$$

In fact, if at least two children contribute, then no more than one stable set can be taken from each child since the children of x are not linked at all. □

Lemma 4.14. *Given a cotree, $MMPS(x)$ and $MMPC(x)$ can be computed for any 0-vertex x in time $\mathcal{O}(h^2)$, where h is the number of children of x.*

Proof. Obviously, we have $MMPS(x) = \cup_i MMPS(c_ix)$, since the union of one stable set from each child yields one stable set and cliques of different children remain disjoint. Also, $MMPC(x)$ is the subgraph realizing the maximum of

$$\big[n(MT(x)), \max_i n(MMPC(c_ix)), \max_{i,j}\big(n(MJS(c_ix)) + n(MC(c_jx))\big)\big].$$

In fact, a maximum $(s,1)$-polar subgraph at a 0-vertex is either a threshold graph having the clique part in one component and the stable set part being the union of maximum stable sets in each child, or the largest maximum $(s,1)$-polar subgraph among the children, or the largest union, among all the children, of a maximum join of stable sets in one child and a

maximum clique in another child (if both are coming from the same child then it amounts to be a maximum $(s,1)$-polar subgraph of the child under consideration).
The time complexity is due to the third term where the maximum is taken over any possible pair of children. □

Lemma 4.15. *Given a cotree, $MP(x)$ can be computed for any 0-vertex x in time $\mathcal{O}(h^2)$, where h is the number of children of x.*

Proof. A maximum polar subgraph is obtained either by taking the union of a maximum polar graph in one child and maximum union of cliques in other children, or by taking the union of the maximum of a threshold graph, a maximum union of cliques and a maximum $(1,t)$-polar subgraph from each child. It follows that $MP(x)$ is the subgraph realizing the maximum of

$$\Big[\max_i\big(n(MP(c_ix)) + \sum_{j\neq i} n(MUC(c_jx))\big),$$
$$\sum_i \max\big(n(MT(c_ix)), n(MUC(c_ix)), n(MMPS(c_ix))\big)\Big].$$

Note that the time complexity is decided by the first term where the maximum is taken on all possible pairs of children. □

Theorem 4.16. *For any cograph G given by its cotree, $MP(G)$ can be computed in time $\mathcal{O}(n^2)$.*

Proof. One may think of an algorithm searching the cotree from the leaves to the root, and computing for each vertex of the cotree a maximum polar subgraph; the one computed at the root provides $MP(G)$. By Lemma 4.15, one can compute a maximum polar subgraph at a 0-vertex. On the other hand, at a 1-vertex x, we know that the complement of the subgraph remaining under this vertex is a disconnected graph which can be represented by a cotree with a root of type 0. Then applying Lemma 4.15 and taking the complement of the resulting subgraph (thus stable sets and cliques interchange roles) gives a maximum polar subgraph at x.
The initialization of this algorithm is done by the following assignments: for a vertex x which is a leaf representing the vertex v, $MUC(x)$ and $MMPC(x)$ are in the form (A,B), where $A=\emptyset, B=\{v\}$. $MJS(x)$ and $MMPS(x)$ are in the form (A,B), where $A=\{v\}, B=\emptyset$ and $MP(x)$ is in one of these forms.
The complexity is $\mathcal{O}(n^2)$ since Lemma 4.14 and Lemma 4.15 are applied for all vertices of the cotree. □

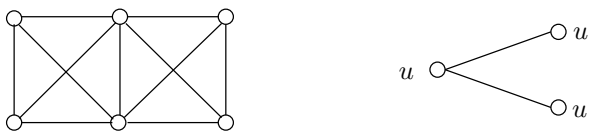
We remark that in a cograph G with weighted vertices, a maximum weight polar subgraph can be found in exactly the same way as previously; it suffices to replace in all lemmas the size of a subgraph by its weight, which is the sum of the weights of the vertices in the subgraph.

4.5 Recognition of polar cographs

Indeed, Theorem 4.16 of Section 4.4 implies a polynomial time recognition algorithm for polar cographs; given a cograph $G = (V, E)$, where $|V| = n$, G is polar if and only if $n(MP(G)) = n$. Here we give a simpler algorithm with a better time complexity deciding whether a given cograph is polar or not, and building a polar partition if there is one. The main idea of the algorithm is that at a 0-vertex of the cotree, the underlying graph is polar if and only if there is a polar partition of each connected component G_i with at most one stable set, or there is at most one connected component with two or more stable sets in every polar partition and all the others are disjoint unions of cliques. First, let us establish the following lemma.

Lemma 4.17. *A connected $(1, \infty)$-polar cograph which is neither a clique nor a threshold graph, is a complete $(1, \infty)$-polar graph $K^*(1, \infty)$. This can be recognized in linear time.*

Proof. Consider a $(1, \infty)$-polar cograph G which is not a clique. If G admits a polar partition with only one clique then it is a threshold graph which can be recognized in linear time; a consequence of Theorem 3.2 is that a cograph is a threshold graph if and only if removing any maximum stable set leaves a clique.
Now, assume that every polar partition of G has more than one clique. Then the stable set is linked to all cliques since G is connected. Moreover, one can verify that the links are complete, otherwise there are P_4's. To recognize $K^*(1, \infty)$, we will repetitively eliminate one of the *real twins*, i.e., adjacent vertices having the same neighborhood, and label the remaining one with u. Note that real twins have to be in a same clique in all polar partitions and they can be found in linear time on the cotree. Consequently, G is a $K^*(1, \infty)$ if and only if at the end of the process of twin elimination, the remaining graph is a complete bipartite graph where all the vertices labeled with u are included in one stable set. We note that this stable set may also contain some non-labeled vertices corresponding to cliques of size one. This later condition is necessary; observe that applying the twin elimination process to the graph of Figure 4.5 a) yields the complete bipartite graph of Figure 4.5 b) but all vertices with label u are not in a same stable set, hence the original graph is not a $K^*(1, \infty)$. Moreover, the polar partition (A, B) is such that A induces the stable set with



a) b)

Figure 4.5: Labeling real twins.

no labeled vertices and the cliques induced by B are obtained by keeping track of the twins (if there is any). Note that this can be done in linear time in the number of vertices. □

In what follows $P(G)$ denotes a polar partition (A, B) of G. Now, if $P(G) = (A, B)$ then we define $\overline{P(G)} = (\overline{B}, \overline{A})$, meaning that stable sets (respectively cliques) of A (respectively B) become cliques (respectively stable sets). Clearly $\overline{P(G)} = P(\overline{G})$ and it is also a polar partition. b is the number of connected components having at least two stable sets in every polar partition. Also, updating (A, B) means that according to the polar partition of the connected component under consideration, we add its stable set(s) in A and its clique(s) in B.

Theorem 4.18. *For any cograph G, Algorithm 9 can recognize whether G is a polar graph in time $\mathcal{O}(n \log n)$ if G.*

Proof. Algorithm 9 gives a negative answer when there is at least one connected component with exactly one stable set in every polar partition and another connected component, which is neither a stable set nor a clique nor a threshold graph nor a $K^*(1, t)$. We know by Lemma 4.17 that these are all possible cases for a $(1, t)$-polar cograph, therefore there is no possible polar partition for such a graph. On the other hand, if G is a polar cograph then Algorithm 9 provides a polar partition.
The complexity is provided by Lemma 4.17 and the fact that these linear operations are repeated at most l times, where l is the height of the cotree, which is $\mathcal{O}(\log n)$. □

Algorithm 9 Polar cograph recognition

Input: a cograph G and its cotree

Output: a polar partition $P(G) = (A, B)$ of G or a negative answer "G is not polar"

1. $A \leftarrow \emptyset$, $B \leftarrow \emptyset$;
2. **if** G is disconnected with components $G_1, \ldots, G_h$ **then**
3. $\quad b \leftarrow 0$;
4. $\quad$ **for** $i = 1$ to h **do**
5. $\quad\quad$ **if** G_i is a stable set or a clique or a threshold graph or a $K^*(1, t)$ for some t **then**
6. $\quad\quad\quad$ update (A, B);
7. $\quad\quad$ **else**
8. $\quad\quad\quad$ $b \leftarrow b + 1$;
9. $\quad$ **if** $b = 0$ **then**
10. $\quad\quad$ **return** $P(G) = (A, B)$.
11. $\quad$ **else if** $b = 1$ and $\exists j$ such that $\forall i \in \{1, \ldots, h\}, i \neq j$ G_i is a clique **then**
12. $\quad\quad$ $P(G) \leftarrow (A, B) \cup \overline{P(\overline{G_j})}$; *//$G_j$ is the only connected component not verifying line 5, hence G_j is not a $(1, t)$-polar cograph*
13. $\quad\quad$ **return** $P(G) = (A, B)$.
14. $\quad$ **else**
15. $\quad\quad$ **print** "G is not a polar graph ", **exit**.
16. **else**
17. $\quad$ $P(G) \leftarrow \overline{P(\overline{G})}$.

4.6 Concluding remarks

In this chapter, we developed a characterization of polar and monopolar cographs by forbidden subgraphs. Polynomial time recognition algorithms for polar and monopolar cographs with time complexities respectively $\mathcal{O}(n \log n)$ and $\mathcal{O}(n)$ are derived, as well as a dynamic programming algorithm to find a maximum (weighted) polar subgraph in a cograph.
Let us recall that although it is $\mathcal{NP}$-complete to recognize polar graphs in general, it becomes polynomially solvable under some circumstances. In fact, we have already pointed out that Theorem 1.34 of Feder et al. (on sparse and dense graphs) implies in Corollary 1.37 that for any graph G and any fixed s and t, it can be determined in polynomial time whether G is (s,t)-polar or not.
There are many questions that still remain to be answered. Among those, the characterization of $(2,2)$-polar cographs by forbidden subgraphs would be a natural continuation. Also one should explore more general subclasses of perfect graphs to characterize their polarity.
This ends the discussion on polar graphs in this dissertation. In the following chapters, we go back to the study of (p,k)-coloring, split-coloring and cocoloring problems in particular classes of graphs; the first candidate is the class of line graphs.

Chapter 5

Line graphs

This chapter is based on [41] which studies the complexity of generalized coloring problems in line graphs.
We show that (p,k)-coloring problems (and hence Min Split-coloring and Min Cocoloring) are polynomially solvable in block graphs which contain line graphs of trees. Then, we detect the first class of graphs where Min Split-coloring and Min Cocoloring behave differently in terms of complexity classes they belong to; in line graphs of bipartite graphs, Min Cocoloring is polynomially solvable while Min Split-coloring is $\mathcal{NP}$-hard. Furthermore, we show that Min Cocoloring remains polynomially solvable in line graphs of line-perfect graphs, whereas it becomes $\mathcal{NP}$-hard in line graphs of bipartite multigraphs even if the multiplicity is 2. Then, we also describe a class of graphs, denoted by $\mathcal{G} = \{G \cup nK_{2n}\}$, where Min Cocoloring is $\mathcal{NP}$-hard and Min Split-coloring is polynomially solvable. This shows that none of these problems is harder than the other one in arbitrary graphs. The results established on generalized coloring problems in this chapter are summarized in Table 5.1 where, for a class of graphs $\mathcal{G}$, the class of line graphs of $\mathcal{G}$ is denoted by $L(\mathcal{G}) = \{L(G) : G \in \mathcal{G}\}$.

	Min Cocol.	**Min Split-col.**	**(p,k)-coloring**
L(tree)	Polynomial	Polynomial	Polynomial
L(bipartite)	Polynomial	$\mathcal{NP}$-hard	$\mathcal{NP}$-hard
L(bipartite multigraphs)	$\mathcal{NP}$-hard	$\mathcal{NP}$-hard	$\mathcal{NP}$-hard
L(line-perfect)	Polynomial	$\mathcal{NP}$-hard	$\mathcal{NP}$-hard
$\mathcal{G} = \{G \cup nK_{2n}\}$	$\mathcal{NP}$-hard	Polynomial	$\mathcal{NP}$-hard

Table 5.1: Complexity of generalized coloring problems in line graphs.

As for the Max (p,k)-colorable subgraph problem in line graphs, we give polynomial time algorithms for finding $\alpha_{0,k}$ in line graphs of line-perfect graphs and finding $\alpha_{1,1}$ in general line graphs. We also analyze the complexity status of finding $\alpha_{p,k}$ in line graphs under the hypothesis that p or k is free or fixed, free meaning that it is not part of the input. These results are summarized in Table 5.2 where we also mention the largest class we know in

which the problem under consideration is polynomially solvable, or the smallest class we know for which the problem under consideration is $\mathcal{NP}$-hard, according to the complexity situation.

(stable sets) k	p **(cliques)**	
	fixed	**free**
0	Polynomial	Polynomial, L(tree) $\mathcal{NP}$-hard, L(bipartite)
1	Polynomial	$\mathcal{NP}$-hard, L(bipartite)
fixed ($k \geq 2$)	Polynomial, L(line-perfect) $\mathcal{NP}$-hard, L(triangle-free)	$\mathcal{NP}$-hard
free	Polynomial, L(line-perfect)	$\mathcal{NP}$-hard

Table 5.2: Max (p, k)-colorable subgraph in line graphs.

5.1 Introduction

Given a graph G, its *line graph*, denoted by $L(G)$, is defined as follows: vertices of $L(G)$ represent edges of G and two vertices of $L(G)$ are adjacent if and only if the corresponding edges are adjacent in G. There is a nice characterization of line graphs by a list of nine forbidden graphs as well as other conditions leading to a linear time recognition (see [17]). Note that line graphs are not perfect in general; a graph is called *line-perfect* whenever its line graph is perfect. Another class of graphs that we make use of in this chapter, is *triangle-free* graphs defined by the absence of induced triangles. Note that bipartite graphs are included in the class of line-perfect graphs (since there is no induced odd cycle of length more than tree nor its complement in line graphs of bipartite graphs) and also in the class of triangle-free graphs, while there is no such a hierarchy between triangle-free graphs and line-perfect graphs.

Clearly, coloring the vertices of a line graph $L(G)$ is equivalent to coloring the edges of G; *Min Edge Coloring* is thus a partition of the edge set of G into a minimum number of pairwise non-adjacent edges, which is called a *matching*.

In the present chapter, we consider (p, k)-colorings of line graphs, but we will formulate them in the original graphs rather than in their line graphs, unless stated otherwise. The (p, k)-coloring of a line-graph $L(G)$ can be viewed as an *edge* (p, k)*-coloring* of G where the objective is to cover the edge set of G by p bundles (*a bundle* is the set of edges adjacent to the same central vertex) and/or triangles (both instead of cliques), and k matchings (instead of stable sets). In this case, we call (p, k)-coloring problems, *edge* (p, k)*-coloring problems*, or equivalently $(p, k)'$*-coloring problems*, and we denote $\alpha_{p,k}(L(G))$ by $\alpha'_{p,k}(G)$. Also, Min Split-coloring in $L(G)$ is called *Min Edge Split-coloring* in G. The objective is to minimize the maximum between the number of triangles or bundles (counted together) and the number of matchings, covering (together) all the edges. The optimal value for

G is $\chi'_S(G) = \chi_S(L(G))$. Analogously, we define *Min Edge Cocoloring* in G as being Min Cocoloring in $L(G)$. Here, we minimize the total number of triangles, bundles and matchings covering all the edges. Then the optimal value of edge cocoloring for G is $z'(G) = z(L(G))$. Although line graphs are not perfect, by considering the edge-versions of generalized coloring problems, we take advantage of the fact that the edges of a graph G can be covered by either $\Delta(G)$ or $\Delta(G)+1$ matchings (where $\Delta(G)$ is the degree of a maximum degree vertex in G) according to Vizing's Theorem [116]; this will be useful to develop polynomial time exact algorithms in the present chapter and approximation algorithms in the next chapter.

5.2 (p,k)-coloring of line graphs

Let us first establish the $\mathcal{NP}$-hardness of both Min Edge Split-coloring and Min Edge Cocoloring in line graphs.

Proposition 5.1. *The following statements hold:*

1. *edge 3-cocolorability is $\mathcal{NP}$-complete;*
2. *edge 3-split-colorability is $\mathcal{NP}$-complete.*

Proof. Statement 1. It is clearly in $\mathcal{NP}$ and we prove its $\mathcal{NP}$-completeness by a reduction from edge 3-colorability (shown to be $\mathcal{NP}$-complete in [88]). Let us consider an instance G of edge 3-colorability. We transform G into an instance $\tilde{G}_C$ of edge 3-cocolorability by adding four disjoint $K_{1,3}$, that is four bundles of size 3 each. Note that in any edge 3-cocoloring of $\tilde{G}_C$, edges of these four bundles have to be covered by three matchings. Consequently, $\tilde{G}_C$ is edge 3-cocolorable if and only if G is edge 3-colorable.
Statement 2. Edge 3-split-colorability is clearly in $\mathcal{NP}$. A similar argument shows that edge 3-colorability also reduces to edge 3-split-colorability. In order to show that, we obtain an instance $\tilde{G}_S$ of edge 3-split-colorability from an instance G of edge 3-colorability by adding three bundles of size 4 each. Then it suffices to observe that in any edge 3-split-coloring of $\tilde{G}_S$, edges of three disjoint $K_{1,4}$ have to be covered by three bundles. This implies that $\tilde{G}_S$ is edge 3-split-colorable if and only if G is 3-edge-colorable. □

In what follows, we consider generalized coloring problems in subclasses of line graphs. In the sequel, an edge cocoloring will be denoted by the set $(\mathcal{Z}', \mathcal{T}')$, where $\mathcal{Z}'$ is the set of vertices (which are the centers of bundles and represent the corresponding bundles in an edge cocoloring) and $\mathcal{T}'$ is the set of triangles, knowing that the remaining edges will be covered by matchings only.

5.2.1 (p,k)-coloring of block graphs

Although we rather study the edge-versions of our problems throughout this chapter, in this particular subsection, we keep our usual problem definitions. Our main purpose is to

consider line graphs of trees but we derive more general results on block graphs. A graph G is a *block graph* if every maximal 2-connected component is a clique. Remark that line graphs of trees are special block graphs where any vertex is contained in at most two cliques; this is due to the fact that the only cliques in a line graph of a tree correspond to bundles in the original graph and one edge belongs to at most two bundles centered at both ends of the edge. In the sequel, we assume that a block graph is given by the list of its cliques (with their sizes) and separation vertices. Note that any block graph can be equivalently given as a forest of block-cutpoint trees where white vertices represent cliques, their weights w represent the sizes of the corresponding cliques and black vertices are the separation vertices. White and black vertices are alternated in the forest; a white vertex representing a clique K is only linked to the black vertices corresponding to separation vertices contained in K. In what follows, we work on this forest representation where any leaf has a unique neighbor which is a black vertex.

Theorem 5.2. *For any block graph, (p_{min}, k)-coloring can be solved in time linear in the number of maximal cliques.*

Proof. Noting that block graphs are perfect, we have to choose a minimum number of cliques such that in the remaining graph, the size of any clique is at most k. To do that, we shall first search the forest once to collect the following data:

- the degree of each vertex;
- the *branch degree* (number of adjacent vertices which are not leaves) of each black vertex;
- the set $\mathcal{L}$ of leaves (white vertices of degree 1 or 0) of weight different from $(k+1)$, or of weight $(k+1)$ if the branch degree of its black vertex is 1.

The solution set $\mathcal{P}$ will contain the vertices representing cliques that should be chosen. We can proceed as in Algorithm 10.

The optimality of the solution results from the following facts:

(a) (expressed by lines 3 to 7 of Algorithm 10) a leaf of weight at least $(k+2)$ belongs necessarily to every solution;

(b) (expressed by lines 8 to 10 of Algorithm 10) if a leaf x of weight at most k belongs to an optimal solution $\mathcal{P}^*$, then there is necessarily a white vertex y of weight greater than or equal to $(k+1)$ in the neighborhood of its black neighbor, and $(\mathcal{P}^* \setminus \{x\}) \cup \{y\}$ is another optimal solution; so we do not introduce such an x into $\mathcal{P}$;

(c) (expressed by lines 11 to 13 of Algorithm 10) if an optimal solution contains a leaf x of weight $(k+1)$ for which a unique vertex y in the neighborhood of its black neighbor is not a leaf, then $(\mathcal{P}^* \setminus \{x\}) \cup \{y\}$ is an optimal solution as well.

Algorithm 10 (p_{min}, k)-coloring of block graphs

Input: the forest representation of a block graph G, set $\mathcal{L}$ of leaves, an integer k

Output: set $\mathcal{P}$ of cliques in a (p_{min}, k)-coloring of G

1. **while** $\mathcal{L} \neq \emptyset$ **do**
2. pick a vertex x in $\mathcal{L}$;
3. **if** $w(x) \geq k+2$ **then**
4. introduce x into $\mathcal{P}$;
5. remove x and its black vertex v;
6. decrease the weight of the neighbors of v by 1;
7. update the degrees of these neighbors;
8. **else if** $w(x) \leq k$ **then**
9. remove x;
10. decrease the degree of its black vertex by 1;
11. **else if** $\omega(x) = k+1$ **then**
12. introduce into $\mathcal{P}$ the unique white vertex y adjacent to its black vertex;
13. delete x, its black vertex, y and all black vertices adjacent to y;
14. **for all** black vertex deleted at step 5 or 13 **do**
15. decrease the weights of its neighbors by 1;
16. update the degrees of its neighbors, the list $\mathcal{L}$ and branch degrees of black vertices adjacent to leaves just introduced into $\mathcal{L}$;
17. delete black vertices of degree 1;
18. **for all** connected component **do**
19. choose exactly one white vertex to be introduced into $\mathcal{P}$;
20. delete the component.

Each time one applies rule (a), (b) or (c), the remaining graph is a forest whose trees have the same black-white structure, and an optimal solution of this graph extends the current solution to an optimal solution (of the whole graph).
Finally, note that when $\mathcal{L}$ becomes empty, the remaining graph consists of independent stars with a black center and white leaves of weight $(k+1)$; choices made at step 19 are obviously optimal.
Let us now evaluate the complexity. The initialization phase needs time $\mathcal{O}(L)$ (searching algorithm), where L denotes the number of white vertices (there are at most L black vertices). At each step 5, 9, 13 at least one white vertex is removed. Let us first evaluate the number of updates in step 16 except for branch degrees. Such updates are performed when a black vertex is removed, and concern directly its neighborhood. The total complexity is $\mathcal{O}(L)$ (the number of edges). Finally, the number of times the branch degree of a vertex is updated is at most the degree of this vertex. Therefore, the overall complexity is $\mathcal{O}(L)$. □

Note that the above result provides polynomial algorithms for all our generalized coloring problems (not necessarily with the same time complexities) in block graphs and consequently in line graphs of trees as well.

5.2.2 Edge cocoloring of bipartite graphs

In this section, we give a polynomial time algorithm for Min Edge Cocoloring in bipartite graphs and we show that Min Edge Cocoloring in bipartite multigraphs become $\mathcal{NP}$-hard even with multiplicity 2.
In bipartite graphs, Min Edge Cocoloring consists of covering the edge set of bipartite graphs with a minimum number of matchings and bundles (since there is no triangle). Knowing that in bipartite graphs, there are always $\Delta(G)$ matchings covering all the edges, at each step of our algorithm, we will decide either to complete the edge cocoloring by matchings only, or to introduce some vertices into the solution. The following lemma gives the decision rule.

Lemma 5.3. *For any bipartite graph G, if $z'(G) = \Delta(G) - t$ where $t > 0$, then vertices with degree strictly greater than $\Delta(G) - t$ are in $\mathcal{Z}'$.*

Proof. Suppose that we do not include in $\mathcal{Z}'$ all vertices with degree greater than $\Delta(G) - t$. Then, there is a vertex with degree at least $\Delta(G) - t + 1 - p$, where p is the total number of vertices introduced into $\mathcal{Z}'$. It is easy to verify that in what follows, we cannot have a solution with a value better than $\Delta(G) - t + 1$. □

This result allows us to derive the following recursive algorithm to find an optimal edge cocoloring in bipartite graphs.

Theorem 5.4. *For any bipartite graph, Algorithm 11 solves Min Edge Cocoloring in time $\mathcal{O}((m+n)\log n)$.*

Algorithm 11 Edge cocoloring of bipartite graphs

Input: a bipartite graph G

Output: a minimum edge cocoloring $\mathcal{Z}'$ of G with value $z'(G)$

1. **if** $G \neq \emptyset$ **then**
2. pick a vertex x of maximum degree in G;
3. $z'(G) = \min\{\Delta(G), 1 + z'(G[V \setminus \{x\}]\}$;
4. **if** $z'(G) = 1 + z'(G[V \setminus \{x\}])$ **then**
5. $\mathcal{Z}' \leftarrow \mathcal{Z}' \cup \{x\}$;
6. **if** $z'(G) = \Delta(G)$ **then**
7. $\mathcal{Z}' \leftarrow \emptyset$.

Proof. The correctness of Algorithm 11 follows from Lemma 5.3: at each step, either all maximum degree vertices have to be included in $\mathcal{Z}'$, or we can complete the edge cocoloring by matchings only, and no other decision would be better.
The time complexity is $\mathcal{O}((m+n)\log n)$ by using a heap with the vertices in non increasing order of degrees. In fact, updating the structure when the vertex on the top of the heap is removed can be made in time $\mathcal{O}(\log n)$ and there are exactly $(m+n)$ updates. □

Note that Algorithm 11 does not give explicitely an optimal edge cocoloring; it only computes its value. The best known algorithm to complete the edge cocoloring of Algorithm 11 by an optimal edge coloring in the remaining graph is due to Schrijver [108]; this would add a term of $\mathcal{O}(\Delta m)$ to the overall time complexity.

Remark 5.5. *If triangles are excluded from any solution, then Algorithm 11 finds an optimal cocoloring in every class of graphs where $\chi'(G) = \Delta(G)$ holds.*

Theorem 5.6. *The problem of deciding whether $z'(G) \leq k$, where $k \in \mathbb{N}$, is $\mathcal{NP}$-complete in bipartite multigraphs, even if the multiplicity is 2.*

Proof. The problem trivially belongs to $\mathcal{NP}$. In order to prove its intractability, we propose a reduction from a restriction of 3-SAT (Lemma 5.7) by revisiting the usual reduction from 3-SAT to vertex covering [70] and then by using a reduction from vertex covering[1] to our edge cocoloring problem.

Lemma 5.7. *The restriction of 3-SAT for which each literal appears at most three times, and the number of clauses is at least 3/2 times the number of variables is $\mathcal{NP}$-complete.*

Proof. Let us consider an instance I of 3-SAT with $n > 6$ variables $(x_1, \ldots, x_n)$ and m clauses $C_1, \ldots, C_m$. Without loss of generality, we can assume that every variable appears at least two times: once in the positive form and once in the negative one. One can also construct (see for instance [106]) an equivalent instance I' for which every variable appears in

[1] Given a graph $G = (V, E)$, a k-vertex cover is a subset $V' \subseteq V$ of size at most k such that for all $v_i v_j \in E$, at least one of v_i and v_j is in V'. Deciding whether G admits a k-vertex cover or not is known to be $\mathcal{NP}$-complete [70].

at most three clauses and every literal in at most two clauses. In fact, if a variable x_i appears k times in I, where $k \geq 4$, we replace x_i by k new variables $x_i^1, \ldots, x_i^k$: the first occurrence of x_i is replaced by x_i^1 (the nature of the related literal, positive or negative, being preserved), the second one by x_i^2, and so on. We introduce k 3-clauses $(\bar{x}_i^1, x_i^2, x_i^2), \ldots, (\bar{x}_i^k, x_i^1, x_i^1)$ guaranteeing that variables $x_i^1, \ldots, x_i^k$ have the same truth value for any feasible truth assignment.
We denote by n' the number of variables in I' and by m' the number of clauses. It is straightforward to verify that in the new instance, each literal appears at most three times, and every variable at most four times and at least twice. Consequently, $m' \geq 2n'/3$ and clauses contain at most $4n'$ literals (with repetitions). It remains possible to use $2n'$ literals $l_1, \ldots, l_{2n'}$ in additional clauses such that every literal appears at most three times. We then add n' new variables $y_1, \ldots, y_{n'}$ and $n'(2 + 2\lfloor n'/3 \rfloor)$ clauses: $(y_i, \bar{y}_i, l_i), (y_i, \bar{y}_i, l_{n'+i})$, $i = 1, \ldots, n'$, and $(y_i, \bar{y}_i, y_{n'-i}), (y_{\lfloor n'/3 \rfloor + i}, \bar{y}_{\lfloor n'/3 \rfloor + i}, \bar{y}_{n'-i})$, $i = 1, \ldots, \lfloor n'/3 \rfloor$. These clauses are true for every truth assignment and every literal appears at most three times among all clauses. Moreover, there are at least $2n'/3 + 2n' + 2\lfloor n'/3 \rfloor > 3n'$ clauses ($n' > 6$) for $2n'$ variables. Note finally that the above construction is polynomial, which concludes the proof of Lemma 5.7. □

Let I be an instance of 3-SAT with n variables and m clauses satisfying the hypotheses of Lemma 5.7: $m \geq 3n/2$ and each literal appears at most three times.
Let us then revisit the reduction to vertex covering [70]. One can construct in polynomial time a graph $G = (V, E)$ where $|V| = 2n + 3m$ and $|E| = n + 6m$ and satisfying $\Delta(G) \leq 4$ such that:

$$\tau(G) \leq k = n + 2m \Leftrightarrow \text{the restricted 3-SAT instance is satisfiable.}$$

Here $\tau(G)$ denotes the value of a minimum vertex cover in G. In what follows, we show how to construct in polynomial time a bipartite multigraph $\mathcal{B}$ with edge-multiplicity not exceeding 2 such that $\tau(G) \leq n + 2m \Leftrightarrow z'(\mathcal{B}) \leq 3n + 6m + 8$.

Construction of $\mathcal{B}$

Let us first consider the bipartite representation of G: B_G is a simple bipartite graph defined by $B_G = (V, E, E_B), ve \in E_B \Leftrightarrow v$ is incident to e in G. In $\mathcal{B}$, every E-vertex has a degree 2 while every V-vertex has a degree not greater than $\Delta(G) \leq 4$. We first add to V a set W of $m + 4$ isolated vertices. For every vertex $e \in E$, we add a set X_e of $2n + 4m + 7$ vertices $X_e = \{x_e^i, i = 1, \ldots, 2n + 4m + 7\}$ all connected to e by a simple edge: $E_e = \{ex_e^i, i = 1, \ldots, 2n + 4m + 7\}$. Finally we add a set $K = \{y_1, \ldots, y_{k+2}\} = \{y_1, \ldots, y_{n+2m+2}\}$ of vertices completely connected to vertices of $V \cup W$ by double edges: let $E_K = \{(y_i v)_1, (y_i v)_2, i = 1, \ldots, n + 2m + 2, v \in V \cup W\}$. Thus:

$$\mathcal{B} = (K \cup E, V \cup W \cup (\cup_{e \in E} X_e), E_B \cup (\cup_{e \in E} E_e) \cup E_K).$$

We denote by $V^1_{\mathcal{B}} = K \cup E$ and by $V^2_{\mathcal{B}} = V \cup W \cup (\cup_{e\in E} X_e)$ the color classes of $\mathcal{B}$. The construction is clearly polynomial. Let us then evaluate the degree of vertices in $\mathcal{B}$:

$$\begin{aligned}
&\forall v \in V \cup W, d_{\mathcal{B}}(v) \leq 4 + 2k + 4 = 2n + 4m + 8;\\
&\forall e \in E, d_{\mathcal{B}}(e) = 2n + 4m + 9;\\
&\forall y \in K, d_{\mathcal{B}}(y) = 2|V \cup W| = 4n + 8m + 8;\\
&\forall x \in \cup_{e\in E} X_e, d_{\mathcal{B}}(x) = 1.
\end{aligned}$$

$\tau(G) \leq n + 2m \Rightarrow z'(\mathcal{B}) \leq 3n + 6m + 8$

Let us now suppose that $\tau(G) \leq k = n + 2m$ and denote by $V' \subset V$, where $|V'| = n + 2m$, a vertex cover of G. Let us then consider $\mathcal{B} \setminus V' = \mathcal{B}[V^1_{\mathcal{B}} \cup V^2_{\mathcal{B}} \setminus V']$.
Since $|V'| = n + 2m$ and V' covers edges of G, when one removes V', the degree of vertices in E decreases at least by 1 and the degree of vertices in K decreases by $2(n + 2m)$. Thus, the maximum degree of $\mathcal{B} \setminus V'$ satisfies:

$$\begin{aligned}
\Delta(\mathcal{B} \setminus V') &\leq \max\{2n + 4m + 8, 2n + 4m + 9 - 1, 4n + 8m + 8 - 2(n + 2m)\}\\
&= 2n + 4m + 8.
\end{aligned}$$

Consequently, $z'(\mathcal{B} \setminus V') \leq n + 2m + 2n + 4m + 8 = 3n + 6m + 8$.

$\tau(G) \leq n + 2m \Leftarrow z'(\mathcal{B}) \leq 3n + 6m + 8$

Let us now suppose $z'(\mathcal{B}) \leq 3n+6m+8$ and consider five sets $X' \subset (\cup_{e\in E} X_e)$, $E' \subset E, V' \subset V, W' \subset W$ and $K' \subset K$, such that:

$$|W'| + |X'| + |E'| + |V'| + |K'| + \Delta(\mathcal{B} \setminus (W' \cup X' \cup E' \cup V' \cup K')) \leq 3n + 6m + 8. \quad (5.1)$$

Let us denote by $\mathcal{B}'$ the graph $\mathcal{B}\setminus(W'\cup X'\cup E'\cup V'\cup K') = \mathcal{B}[V^1_{\mathcal{B}}\cup V^2_{\mathcal{B}}\setminus(W'\cup X'\cup E'\cup V'\cup K')]$.
Without loss of generality, we can assume that $X' = \emptyset$: in fact, if one adds to the remaining graph every vertices of X' and deletes their neighbors, the degree cannot increase while the number of vertices cannot decrease. The same holds for W'.
Let us then notice that $|E'| \leq 2n + 4m + 8 < |E|$. In fact, in the opposite case, we deduce from inequality (5.1) that

$$|K'| + |V'| < 3n + 6m + 8 - (2n + 4m + 8) = n + 2m < |K|.$$

Consequently, $K \setminus K' \neq \emptyset$ and $|(V \cup W) \setminus V'| > n + 2m + 4$, which implies that $\Delta(\mathcal{B}') > 2n + 4m + 8$. Finally, by using $|E'| > 2n + 4m + 8$, we get:

$$|E'| + \Delta(\mathcal{B}') > 4n + 8m + 16 > 3n + 6m + 8$$

which contradicts inequality (5.1).
Since $|E'| < |E|$ and $X' = \emptyset$, we deduce $\Delta(\mathcal{B}') \geq 2n + 4m + 7$ and inequality (5.1) implies:

$$|K'| + |E'| + |V'| \leq 3n + 6m + 8 - (2n + 4m + 7) = n + 2m + 1.$$

We deduce $K \setminus K' \neq \emptyset$ and

$$2(|E \setminus E'|) \geq 2(n + 6m - (n + 2m + 1) + |V'|) = 8m - 2 + 2|V'| > 4|V'|.$$

Consequently, since in B_G the degree of every vertex in V is at most 4 and the degree of vertices in E is 2, at least one vertex in $E \setminus E'$ has a neighbor in $V \setminus V'$, which implies $\Delta(\mathcal{B}') \geq 2n + 4m + 8$.
Then, inequality (5.1) implies $|V'| \leq n+2m$ and moreover, since $K \setminus K' \neq \emptyset$, by considering the degree of vertices in $K \setminus K'$, we get

$$\Delta(\mathcal{B}') \geq 4n + 8m + 8 - 2|V'|.$$

Inequality (5.1) implies $|V'| \geq 4n+8m+8-(3n+6m+8) = n+2m$. Thus $|V'| = n+2m$. Consequently, $|V'| + \Delta(\mathcal{B}') = 3n + 6m + 8$ and then inequality (5.1) implies $|E'| = 0$. Therefore, V' covers every edges of G since in the opposite case $\Delta(\mathcal{B}') \geq 2n + 4m + 9$ and consequently $z'(\mathcal{B}) \geq 3n + 6m + 9$, which concludes the proof. □

5.2.3 Edge split-coloring of bipartite graphs

In this section, we establish the $\mathcal{NP}$-hardness of Min Edge Split-coloring in bipartite graphs. We also describe a class of graphs where Min Split-coloring is polynomially solvable but Min Cocoloring is $\mathcal{NP}$-hard; to our knowledge, this class turns out to be the first one with this property.

Theorem 5.8. *Min Edge Split-coloring is $\mathcal{NP}$-hard in bipartite graphs.*

Proof. The decision version is clearly in $\mathcal{NP}$. In order to prove that it is $\mathcal{NP}$-complete, we propose a reduction from $(p_{min}, 1)'$-coloring in bipartite graphs.
Let (B, k) be an instance of this problem, where $B = (V_B^1, V_B^2, E_B)$ is a bipartite graph and $k \in \mathbb{N}$. The related question is: *is it possible to remove k vertices from B such that the remaining graph has a degree at most 1?* This problem is $\mathcal{NP}$-complete [119].
Without loss of generality, we can assume that $k \leq \tau(B) - 1$ since, in the opposite case, the instance is clearly positive. We construct a bipartite graph B' by adding, for every vertex v of B, $(k-1)$ vertices $(x_v^1, \ldots, x_v^{k-1})$ which are connected to v:

$$B' = (V_B^1 \cup (\cup_{v \in V_B^2} X_v), V_B^2 \cup (\cup_{v \in V_B^1} X_v), E_B \cup E_X) = (V_{B'}^1, V_{B'}^2, E_B \cup E_X)$$

where $\forall v \in V_B^1 \cup V_B^2, X_v = \{x_v^1, \ldots, x_v^{k-1}\}$ and $E_X = \{(v, x_v), v \in V_B^1 \cup V_B^2, x_v \in X_v\}$. Let us also denote $V_B^1 \cup V_B^2$ by V_B, and $V_{B'}^1 \cup V_{B'}^2$ by $V_{B'}$.
Then, we consider $L(B')$ as an instance of split-coloring and show that the instance (B, k) is positive if and only if $\chi_S(L(B')) \leq k$.
Let us first suppose that (B, k) is positive, and consider a set $V' \subset V_B$ such that $|V'| \leq k$ and $\Delta(B[V_B \setminus V']) \leq 1$. Since $k \leq \tau(B) - 1$, we have $\Delta(B[V_B \setminus V']) = 1$. Consequently, by

construction of B', we get $\Delta(B'[V_{B'} \setminus V']) = k$, which implies $\chi_S(L(B')) \leq k$.
Let us now suppose that $\chi_S(L(B')) \leq k$. It means that:

$$\exists V' \subset V_{B'}, |V'| \leq k, \Delta(B'[V_{B'} \setminus V']) \leq k.$$

In fact, an optimal split-coloring of $L(B')$ contains at most k cliques which can be covered by k maximal cliques, corresponding, in B', to vertices in V' (since there is no triangle in B'). Without loss of generality, we can assume $V' \subset V_B$: in the opposite case, consider the set $\big(V' \setminus (V' \cap (\cup_{v \in V_B} X_v))\big) \cup N(V' \cap (\cup_{v \in V_B} X_v))$, where for a set of vertices A, $N(A)$ denotes the set of neighbors of vertices in A. Then, $\Delta(B'[V_{B'} \setminus V']) \leq k \Rightarrow \Delta(B[V_B \setminus V']) \leq 1$, which implies that (B,k) is positive. □

Proposition 5.9. *In the class of graphs $\mathcal{G} = \{G \cup nK_{2n} : G$ is an arbitrary graph of size $n\}$, denoted for short by $\mathcal{G} = \{G \cup nK_{2n}\}$, Min Split-coloring is polynomially solvable while Min Cocoloring is $\mathcal{NP}$-hard.*

Proof. Let us consider the class $\mathcal{G}$ of graphs obtained by adding nK_{2n} to an arbitrary graph G of size n (without any link between G and nK_{2n}). Thus $\forall G' \in \mathcal{G}$, there exists an arbitrary graph G of size n such that $G' = G \cup nK_{2n}$. We also write that $\mathcal{G} = \{G \cup nK_{2n}\}$ by abuse of language. Let us first remark that $\chi_S(kK_k) = z(kK_k) = k$, and furthermore kK_k is k-split-critical but it is not a critical graph for k-cocoloring, i.e., the deletion of any vertex does not decrease its cochromatic number.
As nK_n is an induced structure in G', obviously $\chi_S(G') \geq n$. Now let us consider the split-coloring formed by taking n cliques of size $2n$. Then, the remaining graph G can be partitioned into at most n stable sets. The value of this solution is n and hence $\chi_S(G') = n$.
As for the cocoloring problem in G', note that an optimal cocoloring of G completed by n cliques covering nK_{2n} constitutes a solution and therefore $z(G') \leq z(G) + n$. On the other hand, an optimal cocoloring of G' contains at least n cliques included in nK_{2n}. Consequently, this cocoloring induces a cocoloring of G with at most $z(G') - n$ stable sets or cliques. This implies that $z(G') \geq z(G) + n$ and therefore $z(G') = z(G) + n$.
In conclusion, for any graph G' in $\mathcal{G} = \{G \cup nK_{2n}\}$, Min Split-coloring is trivial while Min Cocoloring is $\mathcal{NP}$-hard. □

5.2.4 Edge cocoloring of line-perfect graphs

Here, the polynomial time algorithm for Min Edge Cocoloring in bipartite graphs is extended to line-perfect graphs. First, we recall some basic properties of line-perfect graphs.

Lemma 5.10. *The following statements are equivalent:*

1. *G is a line-perfect graph;*
2. *G has no induced elementary odd cycles of length more than three [113];*

3. $\forall$ *partial graphs* G' *of* G, $\chi'(G') = \Delta'(G')$, *where* $\Delta'(G)$ *is the largest number of mutually adjacent edges.* □

Min Edge Cocoloring consists in finding a minimum number of bundles, triangles and matchings that cover all the edges. Note that we have $\chi'(G) = \Delta(G)$ if $\Delta(G) \geq 3$ for any line-perfect graph G. It follows that for $\Delta(G) \geq 3$, edges of a line-perfect graph can be polynomially covered by $\Delta(G)$ matchings (considering each color as a matching); an algorithmic proof of it can be found in [35].
Let us first remark that for $\Delta(G) \leq 2$, finding $z'(G)$ is a trivial problem. In fact, $z'(G) = 1$ if $\Delta(G) = 1$ or G is a triangle, $z'(G) = 2$ if G is two triangles, or one triangle and one matching, or two matchings, and $z'(G) = 3$ otherwise. In what follows, we will only deal with the case where $\Delta(G) \geq 3$.
We propose a recursive algorithm which will extend the edge cocoloring with $\Delta(G)$ matchings whenever $\Delta(G) \leq z'(G)$ and in the opposite case, i.e., if $\Delta(G) > z'(G)$, it will include in the solution either a bundle or a triangle in the remaining graph. In what follows, we first show that there exists an optimal edge cocoloring of a line-perfect graph with at most one triangle.

Lemma 5.11. *In a line-perfect graph, if* $(\mathcal{Z}', \mathcal{T}')$ *is an optimal edge cocoloring with a minimum number of triangles, then,*

1. *there are at most two disjoint triangles in* $\mathcal{T}'$;
2. *there are no two triangles with a common vertex in* $\mathcal{T}'$;
3. *there are no two triangles with a common edge in* $\mathcal{T}'$.

Proof. Three triangles can always be replaced by three matchings, and two triangles with at least one common vertex by either two bundles, or a bundle and a matching. Also, two triangles with a common edge can be replaced by either two bundles, or a triangle and a bundle. □

Lemma 5.12. *In a line-perfect graph, there is an optimal edge cocoloring* $(\mathcal{Z}', \mathcal{T}')$ *with at most one triangle.*

Proof. If $\Delta(G) \leq z'(G)$ then an optimal edge cocoloring consists of a usual edge coloring, hence it has no triangle.
If $\Delta(G) > z'(G)$, then in an edge cocoloring $(\mathcal{Z}', \mathcal{T}')$ with a minimum number of triangles, let us observe the two following cases for any vertex x of maximum degree.

Case 1. $x \in \mathcal{Z}' \Longrightarrow z'(G) = 1 + z'(G[V \setminus \{x\}])$.

Case 2. $x \notin \mathcal{Z}' \Longrightarrow$ there is at least one triangle containing vertex x in the solution. According to Lemma 5.11, if we use a triangle containing a vertex x of maximum degree, then we will need at least $d(x) - 2$ bundles and/or matchings in order to

complete the edge cocoloring, implying $z'(G) \geq d(x)-1$ and therefore $z'(G) = d(x)-1$. It means that there is an optimal edge cocoloring with at most one triangle.

□

It follows that in the case where there is an optimal edge cocoloring containing exactly one triangle, this triangle have to contain a vertex x of maximum degree in G. Then, if z'_{TF} stands for the value of an optimal (*triangle-free*) edge cocoloring which does not contain any triangle, then $z'(G) = 1 + z'_{TF}(G(V, E \setminus E(G[T(x)])))$, where $T(x)$ is the set of vertices of the triangle chosen to be in $\mathcal{T}'$.
Bringing all these considerations together, we have

$$z'(G) = \min\{\Delta(G), 1 + z'(G[V \setminus \{x\}]), 1 + z'_{TF}(G(V, E \setminus E(G[T(x)])))\}.$$

Finally, let us note that according to Remark 5.5, z'_{TF} can be computed in the same way as z' in bipartite graphs.

Algorithm 12 Edge cocoloring of line-perfect graphs

Input: a line-perfect graph G

Output: a minimum edge cocoloring $(\mathcal{Z}', \mathcal{T}')$ of G with value $z'(G)$

1. **if** $G \neq \emptyset$ **then**
2. pick a vertex x of maximum degree in G;
3. $z'(G) = \min\{\Delta(G), 1 + z'(G[V \setminus \{x\}]), 1 + z'_{TF}(G(V, E \setminus E(G[T(x)])))\}$, $T(x) = \{x, a, b\}$ such that $(a, b) \in N(x)$ and $(a, b) \in E(G)$;
4. **if** $z'(G) = 1 + z'(G[V \setminus \{x\}])$ **then**
5. $\mathcal{Z}' \leftarrow \mathcal{Z}' \cup \{x\}$;
6. **if** $z'(G) = 1 + z'_{TF}(G(V, E \setminus E(G[T(x)]))$ **then**
7. $\mathcal{T}' \leftarrow \mathcal{T}' \cup \{T(x)\}$;
8. **if** $z'(G) = \Delta(G)$ **then**
9. $\mathcal{Z}' \leftarrow \emptyset, \mathcal{T}' \leftarrow \emptyset$.

Theorem 5.13. *For any line-perfect graph G, Algorithm 12 computes an optimal edge cocoloring $(\mathcal{Z}', \mathcal{T}')$ of value $z'(G)$ in time $\mathcal{O}((m^2 + mn)\log n)$.*

Proof. The correctness of Algorithm 12 is already verified. For the time complexity, the most expensive part is due to the computation of $z'_{TF}(G(V, E \setminus E(G[T(x)])))$ for each triangle linked to a maximum degree vertex. Let us remark that in a line-perfect graph G, the neighborhood of a maximum degree vertex does not contain an induced path on four vertices, because otherwise $L(G)$ would have an odd cycle of length at least five implying that $L(G)$ is not perfect. It follows that the number of triangles sharing a vertex x of maximum degree is limited by $d(x)$. Therefore, we will compute $z'_{TF}(G(V, E \setminus E(G[T(x)])))$ at most m times, which will take time $\mathcal{O}((m^2 + mn)\log n)$. □

5.3 Max (p,k)-colorable subgraph in line graphs

Let us recall that a stable set in a line graph $L(G)$ corresponds to a matching in G, and that a clique in $L(G)$ may correspond either to a triangle or to a bundle (of the same size as the clique) in G. Therefore finding $\alpha_{p,k}(L(G)) = \alpha'_{p,k}(G)$ is equivalent to the problem of finding a subgraph of maximum size (with respect to edges) in G whose edge set can be covered by k matchings and p bundles and/or triangles.

Lemma 5.14. *If $(p,k)'$-coloring problems are $\mathcal{NP}$-hard in a class $\mathcal{G}$ of graphs, then finding $\alpha'_{p,k}$ is also $\mathcal{NP}$-hard in $\mathcal{G}$.* □

5.3.1 Computing $\alpha'_{0,k}$

In this section, we are dealing with the problem of finding a union of k matchings of maximum (total) cardinality in G. First of all, it is well known that for $k = 1$, this problem is polynomially solvable in all finite graphs [48]. Note that finding $\alpha'_{0,k}(G)$ is very closely related to the edge coloring problem since each color in an edge coloring corresponds to a matching. For this reason, the problem of finding a maximum edge subgraph which can be covered by k matchings is called *maximum edge k-coloring* problem [60].
It is shown by Holyer [88] that deciding whether an input graph is edge 3-colorable is $\mathcal{NP}$-complete. This immediately implies that the maximum edge k-coloring problem is $\mathcal{NP}$-hard for every $k \geq 3$. Then, Feige et al. showed that the maximum edge k-coloring is $\mathcal{NP}$-hard even for $k = 2$ [60]. In other words, finding $\alpha'_{0,k}$ in general line graphs is $\mathcal{NP}$-hard for $k \geq 2$. It follows that computing $\alpha'_{p,k}$ is $\mathcal{NP}$-hard for $k \geq 2$ and for all p. In fact, the problem of finding $\alpha'_{0,k}$ in a given graph G can be reduced to the problem of computing $\alpha'_{p,k}$ in a graph G' obtained from G by adding p bundles, each one with n edges where n is the number of vertices in G; any optimal solution giving $\alpha'_{p,k}$ will obviously contain the additional p bundles since no other choice can contribute to the solution more than $n-1$ edges per bundle.

Proposition 5.15. *For any $k \geq 2$, the problem of finding $\alpha'_{0,k}$ is polynomially solvable in line-perfect graphs.*

Proof. Let G be a line-perfect graph. If $\Delta(G) = 2$, then G consists of isolated cycles and paths, and it is trivial to compute $\alpha'_{0,k}$. Otherwise, if G is simple with $\Delta(G) \geq 3$ then $\chi'(G) = \Delta'(G) = \Delta(G)$; therefore, edges of G can be covered by $\Delta(G)$ matchings. Consequently, for $k \geq 3$, the problem of finding $\alpha'_{0,k}$ is equivalent to the (edge) *maximum degree bounded subgraph* problem where given a graph, the objective is to find a subgraph containing as many edges as possible and such that the degree of any vertex is at most k. This problem is known to be polynomially solvable [49] (but $\mathcal{NP}$-hard if the subgraph is required to be connected [70]) and efficient algorithms have been given to find such subgraphs [68]. On the other hand, finding a maximum subgraph with degree bounded by 2 is not equivalent to computing $\alpha'_{0,2}$, since the maximum degree in an odd cycle is bounded by 2

but it can not be covered by 2 matchings. If G is line-perfect, then G does not contain any odd cycle except triangles [113]. Moreover, for $k = 2$, one can find in polynomial time a maximum triangle-free subgraph with degree bounded by 2 using the algorithm designed by Hartvigsen [84]. □

Note that for $k \geq 3$, the problem of finding $\alpha'_{0,k}$ is polynomially solvable in any graph G having $\chi'(G) = \Delta(G)$. As for the bipartite graphs, $\alpha'_{0,k}(G)$ is obtained by a simple maximum flow algorithm (from the source to the sink) on the network containing graph $G = (A, B, E)$ and where the edges between A and B have capacity one, while edges connecting A-vertices to the source and B-vertices to the sink have capacity k. Clearly, the value of a maximum flow from the source to the sink in such a network gives $\alpha'_{0,k}(G)$.
Finally, according to [31], determining whether a graph $G = (V, E)$ contains a 2-regular partial subgraph[2] $G' = (V, E'), E' \subseteq E$ (on the same vertex set) without cycles of length 5 is $\mathcal{NP}$-complete. It follows directly that finding $\alpha'_{0,2}$ is $\mathcal{NP}$-hard already in triangle-free graphs.

5.3.2 Computing $\alpha'_{k,k}$

Let us first consider the case $k = 1$. If $M(G)$ denotes the size of a maximum matching in G, then clearly, we have $M(G) + \Delta(G) - 1 \leq \alpha'_{1,1}(G) \leq M(G) + \Delta(G)$. We will now examine different cases with respect to the value of the maximum degree.
If $\Delta(G) \geq 4$, then let us consider the graph G' obtained from G by linking every maximum degree vertex to an additional vertex v. We have the two following (exclusive) cases.

Case 1. There is a maximum matching in G which leaves at least one maximum degree vertex unsaturated and then we have $M(G) = M(G') - 1$ yielding $\alpha_{1,1}(G) = M(G) + \Delta(G) = M(G') + \Delta(G) - 1$.

Case 2. All maximum matchings in G saturate all maximum degree vertices and then we have $M(G) = M(G')$ yielding $\alpha_{1,1}(G) = M(G) + \Delta(G) - 1 = M(G') + \Delta(G) - 1$.

Consequently, in both cases we have $\alpha'_{1,1}(G) = M(G') + \Delta(G) - 1$.
If $\Delta(G) = 3$, then $\alpha'_{1,1}(G) = M(G) + \Delta(G)$ if and only if there is a maximum matching leaving a maximum degree vertex unsaturated or leaving a triangle free, i.e., containing no edge of a triangle and otherwise $\alpha'_{1,1}(G) = M(G) + \Delta(G) - 1$. Remark that triangles are either single (do not touch any other triangle) or in diamonds (two triangles sharing one edge). Let us construct the graph G' mentioned above and give weights to its edges in the following way: edges of each one of the t_1 single triangles of G have weight 1 (but not the ones created by the addition of v) and all the other edges have weight 0. Note that any matching can have at most one (weighted) edge in each triangle. Now, let us compute a maximum matching M^* of minimum weight $W(M^*)$ in G' and consider the two following (exclusive) cases.

[2] That is a partial subgraph where the degree of each vertex is exactly 2.

Case 1. There is a maximum matching in G which leaves at least one maximum degree vertex unsaturated and then we proceed as in Case 1 for $\Delta(G) \geq 4$.

Case 2. All maximum matchings in G saturate all maximum degree vertices (included those in diamonds, which implies that there is no free triangle in any diamond); consequently $M(G) = M(G')$, then,

i) if $W(M^*) < t_1$, then M^* leaves at least one triangle free and therefore $\alpha'_{1,1}(G) = M(G') + \Delta(G)$;

ii) otherwise every maximum matching has at least one edge of each triangle and therefore $\alpha'_{1,1}(G) = M(G') + \Delta(G) - 1$.

Finally, if $\Delta(G) = 2$, then either G contains a (isolated) triangle and then $\alpha'_{1,1} = M(G)+2$, or there is no triangle in G and in this case we have $\alpha'_{1,1} = M(G) + 1$ if G admits a perfect matching, that is a matching saturating all the vertices, and $\alpha'_{1,1} = M(G) + 2$ otherwise. Note that the result for $\Delta(G) \geq 4$ holds also for bipartite graphs without restriction on maximum degree since there is no triangle in this case.

Theorem 5.16. *$\alpha'_{1,1}(G)$ can be polynomially computed in any graph with the same time complexity as for a maximum matching of minimum weight.*

Proof. One of the above methods has to be applied with respect to the value of $\Delta(G)$ for the graph in consideration. The time complexity is dominated by the case $\Delta(G) = 3$ where we compute a maximum matching of minimum weight in G'. This requires the detection of all triangles in G, which can be done in time $\mathcal{O}(m)$ by a breadth first search since, in the case where $\Delta(G) = 3$, the number of triangles is at most n (this value is attained with $n/4$ isolated cliques of size 4). □

Note that for k not part of the input, Theorem 5.8 together with Lemma 5.14 imply that finding $\alpha'_{k,k}$ is $\mathcal{NP}$-hard even in bipartite graphs.

5.3.3 Computing $\alpha'_{p,0}$

Let us first remark that for a fixed p, an exhaustive search will give $\alpha'_{p,0}$ for an arbitrary graph since the number of maximal cliques is polynomially bounded in line graphs of general graphs. This implies that the complexity status of finding $\alpha'_{p,k}$ for a fixed p is the same as for $\alpha'_{0,k}$. Consequently, finding $\alpha'_{p,k}$ for a fixed p is $\mathcal{NP}$-hard in general graphs while it is polynomial in bipartite graphs and line-perfect graphs. Obviously, this argument is no longer valid for a fixed k and $p = 0$, because the number of maximal stable sets is not necessarily polynomially bounded in line graphs of general graphs; in fact a complete bipartite graph has a non-polynomial number of maximal matchings.
Let us now consider line graphs of triangle-free graphs where cliques correspond only to bundles in the original graph since there is no triangle in triangle-free graphs. Hence finding $\alpha'_{p,0}$ is reduced to searching p bundles containing a maximum number of edges. In addition

to this, we know that Max Stable Set and consequently[3] Min Vertex Cover (the problem of covering all edges by a minimum number of vertices; a vertex v is said to cover an edge e if e contains v as an endpoint) are $\mathcal{NP}$-hard in the class of triangle-free graphs [107]. It is straightforward to see that computing $\alpha'_{p,0}$ is also $\mathcal{NP}$-hard because otherwise we could find an optimal vertex covering by checking whether $\alpha'_{p,0}(G) = m$ or not for different values of p.
On the other hand, the problem of finding $\alpha'_{p,0}$ is $\mathcal{NP}$-hard in bipartite graphs but polynomially solvable in trees [6].
Finally, let us recall that, as mentioned in Section 5.2.3, $(p,1)'$-coloring problem (for p free) is $\mathcal{NP}$-hard in bipartite graphs. Lemma 5.14 implies that finding $\alpha'_{p,1}$ is $\mathcal{NP}$-hard for bipartite graphs and consequently for line-perfect graphs. Furthermore, the same reduction as in the proof of Theorem 5.8 implies that finding $\alpha'_{p,k}$ remains $\mathcal{NP}$-hard for any fixed k.
The results of this section, namely the complexity status of finding $\alpha'_{p,k}$ according to p and k in the class of bipartite graphs, line-perfect graphs and triangle-free graphs, are summarized in Table 5.2 at the beginning of the present chapter.

5.4 Conclusion

This research has settled the complexity status of generalized vertex coloring problems in line graphs. The next section will complete these results by developing approximation algorithms for $\mathcal{NP}$-hard cases occurring in line graphs (and also in comparability graphs and in arbitrary graphs).
A natural question arising from this chapter concerns the relative difficulties of Min Split-coloring and Min Cocoloring. Here we emphasized the existence of classes of graphs, namely line graphs of bipartite graphs and line graphs of line-perfect graphs, where Min Cocoloring is polynomially solvable whereas Min Split-coloring is $\mathcal{NP}$-hard. One may wonder whether there are known classes of graphs where the converse holds. Our result showing that in the class of graphs $\mathcal{G} = \{G \cup nK_{2n}\}$, Min Cocoloring is $\mathcal{NP}$-hard, while Min Split-coloring is polynomially, solvable indicates that this is not impossible. In the next chapter, we progress in the direction of answering this open question by showing that we can exclude from our research a rather large class of graphs where Min Split-coloring is at least as difficult as Min Cocoloring. More results on this topic are then obtained in Chapter 7.

[3] According to a well known result [70], in any graph $G = (V, E)$, a set $V' \subseteq V$ is a stable set if and only if $V \setminus V'$ is a vertex cover, i.e., a set of vertices covering all the edges.

Chapter 6

Approximation and bounds

In this chapter, based on [39], we derive approximation results for Min Split-coloring and Min Cocoloring in line graphs, comparability graphs and general graphs. To our knowledge, this provides the first approximation results for Min Split-coloring.
We first consider the class of line graphs. In the previous chapter, we showed that Min Split-coloring is $\mathcal{NP}$-hard in line graphs of bipartite graphs while Min Cocoloring is polynomial for this class. Here we approximate Min Split-coloring and Min Cocoloring in line graphs with approximation ratios of 7/3 and 2 respectively. Then, we give improved approximations of Min Split-coloring in line graphs of line-perfect graphs (with a ratio of 2) and line graphs of bipartite graphs (with a ratio of 1.78).
In addition, noticing that Min Split-coloring is $\mathcal{NP}$-hard in comparability graphs, also known as transitively orientable graphs, we give a 2-approximation algorithm for this case; this result is the split counterpart of a result for cocoloring in comparability graphs [65]. In fact, the $\mathcal{NP}$-hardness of Min Split-coloring in comparability graphs follows from a more general result that we derive: Min Cocoloring reduces to Min Split-coloring in the class of graphs closed under addition of disjoint cliques without link to the rest of the graph and under addition of a complete k-partite graph completely linked to the rest of the graph. Note that this class strictly contains the class of perfect graphs.
We also study the standard and differential approximation behavior of Min Cocoloring and Min Split-coloring in general graphs. From the standard approximation point of view, negative approximability results of Min Coloring hold also for Min Split-coloring and Min Cocoloring. In return, we show that Min Split-coloring and Min Cocoloring are better approximable than Min Coloring in terms of differential approximation ratio since they admit a polynomial time differential approximation scheme, i.e., a $(1-\epsilon)$-differential approximation algorithm with complexity $\mathcal{O}(n^{1+3/\epsilon})$ for every ϵ such that $0 < \epsilon < 1$, and Min Coloring admit only a constant differential ratio. On the other hand, a fully polynomial time differential approximation scheme (the same ratio with complexity polynomial in $1/\epsilon$) cannot be guaranteed for any of them, unless $\mathcal{P}$=$\mathcal{NP}$.
Let us state in Table 6.1 the complexity and approximation results obtained in this chapter; references are given whenever the results were known before. A "-" in an entry indicates

that the corresponding problem has no meaning. Recall that $\mathcal{G} = \{G \cup nK_{2n}\}$, already introduced in the previous chapter, is the class of graphs obtained by taking any arbitrary graph of size n and adding n disjoint cliques of size $2n$ each.

Class of gr.		Complexity	Approx.	Non-approx.
L(G)	χ_S	$\mathcal{NP}$-hard	7/3	4/3-ϵ if $\mathcal{P}\neq\mathcal{NP}$
	z	$\mathcal{NP}$-hard	2	
L(line -perfect)	χ_S	$\mathcal{NP}$-hard [41]	2	DFPTAS
	z	$\mathcal{O}((m^2+mn)\log n)$ [41]	-	-
L(Bipart.)	χ_S	$\mathcal{NP}$-hard [41]	1.78	DFPTAS
	z	$\mathcal{O}((m+n)\log n)$ [41]	-	-
Compar.	χ_S	$\mathcal{NP}$-hard	2	DFPTAS
	z	$\mathcal{NP}$-hard [117]	1.71 [65]	DFPTAS
$\mathcal{G} =$ $\{G \cup nK_{2n}\}$	χ_S	$\mathcal{O}(1)$ [41]	-	-
	z	$\mathcal{NP}$-hard [41]	3/2	DFPTAS
General	χ_S	$\mathcal{NP}$-hard [19]	DPTAS[a]	$n^{1/14-\epsilon}$ if $\mathcal{P}\neq\mathcal{NP}$
	z	$\mathcal{NP}$-hard [19]		$n^{1/2-\epsilon}$ if co$\mathcal{RP}\neq\mathcal{NP}$ DFPTAS

[a] Obviously, this result also holds for all subclasses of finite graphs.

Table 6.1: Approximation results for Min Split-coloring and Min Cocoloring.

Finally, we derive some bounds on optimal split-colorings and cocolorings making use of some intuitive ideas as well as by generalizing some well known sequential usual coloring algorithms, namely the one due to Welsh-Powell [118] and Matula's smallest last algorithm [101, 102].

6.1 Approximation theory

As soon as the $\mathcal{NP}$-completeness theory was established, the research community started to explore different ways of coping with $\mathcal{NP}$-complete problems. Researchers having rather an intuition that $\mathcal{P} \neq \mathcal{NP}$ but not being able to prove it, tried to provide at least some "good" solutions for hard optimization problems. Some of them adopted the approach of developing intuitively good strategies to find solutions "close" to the optimal one without having any theoretical guarantee on their distance to the optimum but satisfied by their practical quality. The resulting procedures, known as *heuristics*, have the advantage of providing solutions that can be used in real world applications since relatively "good" solutions are obtained in some "reasonable" time. For instance, tabu search, simulated annealing and genetic algorithms are some of the widely studied heuristics which are adapted with a great success to a broad variety of problems. See [1] and [122] for a general description of these heuristics as well as more technical information and references on each of them. Never-

theless, the power of heuristics is measured mainly by empirical comparisons[1]. Although problem specific fine tunings often produces satisfactory solutions, they are not generally built according to an a priori analysis to predict their performance. In other words, they provide solutions that can be used in practice, but they do not give a deeper insight on the intrinsic difficulties of problems.

In order to learn more on the hierarchy of the $\mathcal{NP}$-complete problems, another approach, called *approximation theory*, was developed. This allows to say, for a given approximation algorithm, how far can be an approximate solution from an optimal one in the worst case[2]. Here, the objective is to provide polynomial time approximation algorithms with a performance guarantee that a solution obtained by that algorithm is never more distant from the optimum than by a prespecified measure. A commonly accepted measure is the approximation ratio that we introduce below. Note that this type of algorithms can be polynomial with a high degree, meaning that they may be unusable in practice because of the time requirements. However, this theory allows us to build a hierarchy between $\mathcal{NP}$-hard problems according to the class of approximability they belong to. In fact, approximation algorithms are classified according to the type of the approximation ratio they can guarantee (see the next subsection for their definitions). Then, one can establish positive or negative results saying that a problem admits an approximation algorithm of certain type, or respectively that it does not admit an approximation ratio of certain type. Consequently, this gives a hierarchy of problems rather "well" approximable or not. This is the approach we adopt in this chapter.

Note that the notion of "good" approximability is closely related to the approximation ratio we use. In the sequel, we define two of them, namely standard and differential approximation ratios. Works in this context have pointed out that it is often interesting to simultaneously consider both points of view since these ratios provide different pieces of information about combinatorial problems [43, 52].

6.1.1 Standard approximation ratio

We define the classical approximation ratio as follows.

Definition 6.1 (Standard approximation ratio). *Let Π be an optimization problem and I an instance belonging to the set $\mathbb{I}_\Pi$ of instances of Π. The standard approximation ratio of an algorithm $\mathcal{A}$ and for I is defined by*

$$\rho_{\mathcal{A}}(I) = \frac{\lambda(I)}{\beta(I)}$$

[1] See Chapter 2 of [1] by M. Yannakakis for the foundations of the complexity theory of local search. The complexity class $\mathcal{PLS}$, short for polynomial-time local search, is introduced to capture the problems whose neighborhood can be searched in polynomial time. In spite of these theoretical developments, building heuristics remain an experimental art.

[2] Average case analysis do also exist in the literature, see for instance Chapter 9 of [7] or Chapter 3 of [1]. It is based on an analysis of the expected behavior of approximation algorithms under particular probability distributions, and leads, in general, to a better approximation behavior compared to the worst case analysis.

where $\lambda(I)$ is the value of the solution of I given by $\mathcal{A}$ and $\beta(I)$ is the value of an optimal solution.

The absolute constant approximation ratio $\rho_{\mathcal{A}}$ of an algorithm $\mathcal{A}$ for a minimization (respectively, maximization) problem is then determined by the worst case scenario:

$$\rho_{\mathcal{A}} = \sup_{I \in \mathbb{I}_{\Pi}} \rho_{\mathcal{A}}(I) \qquad \text{(respectively, } \rho_{\mathcal{A}} = \inf_{I \in \mathbb{I}_{\Pi}} \rho_{\mathcal{A}}(I)\text{)}.$$

Note that we simply denote by ρ the absolute approximation ratio since the algorithm under consideration is always clear from the context.
In our case, if some ambiguity arises, we write λ_S, β_S (respectively λ_C, β_C) in order to refer to Min Split-coloring (respectively Min Cocoloring).
As we have already mentioned, different behavior of $\mathcal{NP}$-hard optimization problems can be enlightened by means of approximation classes. We define some classes of optimization problems sharing similar approximability properties. If $\mathcal{P} \neq \mathcal{NP}$, they form a strict hierarchy from badly to rather well approximated problems. See [7] for more information on approximation classes.

Definition 6.2 (APX). *APX is the class of $\mathcal{NP}$-hard optimization problems such that for some $r \geq 1$, they admit a polynomial time r-approximation algorithm.*

Definition 6.3 (PTAS). *Let Π be an $\mathcal{NP}$-hard optimization problem. An algorithm $\mathcal{A}$ is said to be a polynomial time approximation scheme (PTAS) for Π if, for any instance I of Π and for all $\epsilon > 0$, $\mathcal{A}$ when applied to input (I, ϵ) returns a $(1+\epsilon)$-approximate solution of I in time polynomial in the size of the instance I. PTAS is the class of $\mathcal{NP}$-hard optimization problems admitting a polynomial time approximation scheme.*

Definition 6.4 (FPTAS). *Let Π be an $\mathcal{NP}$-hard optimization problem. An algorithm $\mathcal{A}$ is said to be a fully polynomial time approximation scheme (FPTAS) for Π if, for any instance I of Π and for all $\epsilon > 0$, $\mathcal{A}$ when applied to input (I, ϵ) returns a $(1+\epsilon)$-approximate solution of I in time polynomial both in the size of the instance I and in $1/\epsilon$. FPTAS is the class of $\mathcal{NP}$-hard optimization problems admitting a fully polynomial time approximation scheme.*

6.1.2 Differential approximation ratio

Assume that we have an approximation algorithm for Min Vertex Cover with an approximation ratio of 2. For an instance of size 1000 with an optimal solution value of 490, this algorithm will return a solution of at most 980 vertices. Using the well known result [70] on the links between Min Vertex Cover and Max Stable Set, we can transform every solution for Min Vertex Cover into a solution for Max Stable Set just by taking its complementary in the given graph, and vice versa. This transformation yields a ratio for Max Stable Set much larger than the one for Min Vertex Cover : $\frac{1000-490}{1000-980} = 25.5$. This is the result of the fact that this transformation which preserves the optimal solutions does not preserve the approximation ratio.

Note that the preservation of the approximation ratio by some very "natural" transformations as in the example above, is a very natural property that an approximation ratio should require in order to make the approximability theory compatible with the optimization theory. To this end, we adopt the differential approximation ratio which is defined in an axiomatic way in [44] to allow all the affine transformations of a problem to be equivalent in the sense of approximability. Let us introduce the definition of the differential approximation ratio which takes into account the *worst solution*, obtained by solving the same problem, keeping the same constraints, but with a maximization (respectively minimization) objective function if the original objective is to minimize (respectively maximize)[3]. The value of a worst solution for an instance I, denoted by $\omega(I)$, is sometimes trivial to compute; for instance, for a Min Coloring instance I with n vertices, we have $\omega(I) = n$. The exact definition of $\omega(x)$ has to be specified for each problem.

Definition 6.5 (Differential approximation ratio). *We call the differential approximation ratio, for an instance I of an optimization problem Π and for an approximation algorithm $\mathcal{A}$, the ratio*

$$\delta_{\mathcal{A}}(I) = \frac{|\omega(I) - \lambda(I)|}{|\omega(I) - \beta(I)|}$$

where $\omega(I)$, $\beta(I)$ and $\lambda(I)$ are respectively the values of the worst solution, an optimal solution and the approximate one given by $\mathcal{A}$.

Since the differential constant approximation ratio of an algorithm reflects the worst case approximability, it is defined by

$$\delta_{\mathcal{A}} = \inf_{I \in \mathbb{I}_{\Pi}} \delta_{\mathcal{A}}(I).$$

As previously, we simply write δ where no confusion arises.
Note that $\delta \in [0, 1]$ and the larger the ratio is, the better, without distinction between maximization and minimization problems. Roughly speaking, this ratio gives the position of the approximated value between the worst and the best one.
All the approximability classes introduced previously are similarly defined in the case of differential framework by simply replacing the standard ratio by the differential ratio, and by taking $(1-\epsilon)$ instead of $(1+\epsilon)$ in the definitions of PTAS and FPTAS; they are respectively denoted by DAPX, DPTAS and DPFTAS.
This ratio, also called z-approximation ratio [85], has been used since a long time (see for instance [121]) and is extensively discussed in [44] and [43]; many studies in this area have pointed out that both ratios are complementary without trivial links between them, which emphasizes the interest to systematically study a problem by with both ratios (see [52] and [43]).
In particular, it has the advantage of respecting some affine equivalence such as the equivalence between Max Stable Set and Min Vertex Cover, while both problems are known to

[3] See [44, 37] for a discussion on the worst solution.

have radically different approximation behaviors for the usual ratio[4]. For instance, Min Coloring admits constant differential approximation algorithms, the best ratio currently known being 59/72 [47], while it is hard to approximate from the usual ratio framework (see the inapproximability results in [7]). On the other side, it does not admit any differential PTAS, unless $\mathcal{P}$=$\mathcal{NP}$ ([8]). On the contrary, some other problems are constant approximated with the usual ratio and hard to approximate from the differential point of view and, finally, some problems have similar behavior from both points of view.
For Min Split-coloring, we consider $\lceil n(G)/3 \rceil$ as worst value since one can always assume that each color (except at most one) contains at least three vertices (every set of three vertices induces in G a split graph). The ratio associated with G is $\delta(G) = [\lceil n(G)/3 \rceil - \lambda(G)]/[\lceil n(G)/3 \rceil - \chi_S(G)]$. Similarly, the differential ratio for Min Cocoloring is $[\lceil n(G)/2 \rceil - \lambda(G)]/[\lceil n(G)/2 \rceil - z(G)]$ since any pair of vertices forms either a clique or a stable set. Note that a larger worst value such as n could be also used, leading to better approximation ratios. But it is reasonable to consider the more restrictive values $\lceil n(G)/3 \rceil$ and $\lceil n(G)/2 \rceil$, respectively, in order to avoid an artificial increase of the final ratio (see [44] where the notion of worst value is discussed). It simply corresponds to restrict the analysis to "reasonable" solutions.
It is an easy task to verify the following useful property of differential approximation ratio.

Property 6.6. *The differential ratio is an increasing function with respect to the value of the worst solution.*

In what follows, unless otherwise stated, approximation ratio stands for standard approximation ratio. The differential ratio will be exclusively used in Section 6.5.2 of this chapter and in Section 7.2.4 of Chapter 7.

6.2 Preliminary remarks

Let us first mention the following preliminary result on approximation dealing with standard approximation ratio.

Proposition 6.7. *There is a reduction which preserves approximation between Min Split-coloring and Min Cocoloring: every r-approximation algorithm for one of these problems gives a $2r$-approximation algorithm for the other one.*

Proof. Suppose we have an r-approximation algorithm for Min Cocoloring giving a solution of value $\lambda_C(G)$ for any graph G. Consider the vertex partition of that solution as a split-coloring of value $\lambda_S(G)$. Clearly, we have $\lambda_S(G) \leq \lambda_C(G) \leq rz(G) \leq 2r\chi_S(G)$ since a minimum split-coloring of G provides a cocoloring of value $2\chi_S(G)$.

[4] They both admit an approximation ratio which is infinity. However, more refined analyses give approximation ratios depending on the size of the instance. Given a graph G with n vertices, the best known results state that Max Stable Set is approximable within $\mathcal{O}(\frac{n}{(\log n)^2})$ (see [14]) and Min Vertex Cover is approximable within $(2 - \Theta\frac{1}{\sqrt{\log n}})$ (see [94]).

Similarly, if we have an r-approximation algorithm for Min Split-coloring giving a solution of value $\lambda_S(G)$ for any graph G, then the value of a cocoloring derived from that solution verifies $\lambda_C(G) \leq 2\lambda_S(G) \leq 2r\chi_S(G) \leq 2rz(G)$. □

Corollary 6.8. *For every class of graphs for which $z(G)$ (respectively $\chi_S(G)$) can be computed in polynomial time, Min Cocoloring (respectively Min Split-coloring) induces a 2-approximation for Min Split-coloring (respectively Min Cocoloring).*

It follows that $z(G)$ can be polynomially approximated within a factor of 2 in the class of graphs $\mathcal{G} = \{G \cup nK_{2n}\}$. In fact, a better approximation ratio can easily be obtained. It is shown in Proposition 5.9 that for any $G' \in \mathcal{G}$, we have $z(G') = n + z(G)$, where $G' = G \cup nK_{2n}$ with G of size n. Therefore, $z(G') \geq n + 1$ (since $n \geq 1$ and there is no stable set contained in nK_{2n} in an optimal cocoloring of G') and a cocoloring of value $\lambda_C(G') \leq (3n/2) + 1$ can easily be obtained by taking a solution on G of value $\lceil n/2 \rceil$ (since any pair of vertices forms either a clique or a stable set) and n cliques covering nK_{2n}. This provides an approximation ratio of $3/2$.

6.3 Line graphs

In this section, we give approximation results for Min Split-coloring and Min Cocoloring in line graphs. For this, we use the terminology defined in the previous chapter for generalized colorings of line graphs.
Recall that both edge 3-split-colorability and edge 3-cocolorability are shown to be $\mathcal{NP}$-complete in general line graphs (see Proposition 5.1). Since both Min Edge Split-coloring and Min Edge Cocoloring have integral values, we can immediately deduce the following result.

Corollary 6.9 (of Proposition 5.1). *Both Min Edge Split-coloring and Min Edge Cocoloring are not approximable within a factor of $4/3 - \epsilon$, unless $\mathcal{P}$=$\mathcal{NP}$.*

Furthermore, we showed in Chapter 5 that in line-perfect graphs, Min Edge Cocoloring is polynomially solvable while Min Split-coloring is $\mathcal{NP}$-hard (see Theorems 5.8 and 5.13). Corollary 6.8 combined with these results allows us to state the following approximation result.

Corollary 6.10. *Min Edge Cocoloring provides a 2-approximation for Min Edge Split-coloring in line-perfect graphs in time $\mathcal{O}((m^2 + mn)\log n)$.* □

Indeed, an optimal edge cocoloring of an instance is a 2-approximation of the same instance, now viewed as an instance of Min Edge Split-coloring.
It can be easily observed that this bound of 2 is tight for the graph $G = pK_{2p} \cup pK_p$, which is obviously the line graph of a line-perfect graph. More precisely, we have $z(G) = 2p$ by taking $2p$ (disjoint) cliques. This solution induces a split-coloring of value $2p$ as well.

Nevertheless, we have $\chi_S(G) = p$ by choosing p cliques of size $2p$ and p stable sets covering the remaining p cliques of size p each.
In what follows, we develop an approximation algorithm for Min Edge Split-coloring. Let $\mathcal{A}$ be a polynomial time algorithm computing an edge $(\Delta + 1)$-coloring for any graph of maximum degree Δ [104] and an optimal edge coloring for line-perfect graphs [35]. Also, we denote an edge split-coloring by $\mathcal{S}'$; a solution is the set of edges incident to vertices in $\mathcal{S}'$ completed by an edge coloring. We consider the following algorithm for Min Edge Split-coloring.

Algorithm 13 Greedy edge split-coloring
Input: a graph G, edge coloring algorithm $\mathcal{A}$
Output: an edge split-coloring $\mathcal{S}'$ of G
1. $\mathcal{S}' \leftarrow \emptyset$;
2. **while** $|\mathcal{S}'| < \Delta(G)$ **do**
3. pick a vertex x of maximum degree in G;
4. $\mathcal{S}' \leftarrow \mathcal{S}' \cup \{x\}$;
5. $G \leftarrow G[V \setminus \{x\}]$;
6. compute an edge coloring of the remaining edges by $\mathcal{A}$.

The idea is that, if $k = \min\{d : |\{x : d(x) > d\}| \leq d\}$, then by removing all vertices of degree greater than k (the maximum degree is at most k in the remaining graph) and by completing the solution by $k + 1$ matchings [104], one finds an edge split-coloring of value $k + 1$.

Theorem 6.11. *For any graph G,*

1. *Algorithm 13 computes an edge split-coloring of cardinality at most $2\chi'_S(G) + 1$ in polynomial time.*
2. *Algorithm 13 is a polynomial time 7/3-approximation for Min Edge Split-coloring.*
3. *Algorithm 13 is a polynomial time 2-approximation for Min Edge Split-coloring in line-perfect graphs.*

Proof. Let us consider a graph $G = (V, E)$, it is straightforward to verify that Algorithm 13 computes a split-coloring of G; we denote by $\lambda_{Gr}(G)$ its value. Let $k = \min\{d : |\{x : d(x) > d\}| \leq d\}$. In what follows, we show that $\lambda_{Gr}(G) \leq k + 1 \leq 2\chi'_S(G) + 1$.
Statement 1. Let us first note that if $\chi'_S(G) = 1$, then $\lambda_{Gr}(G)$ is either 1 or 2; on the other hand, if $\chi'_S(G) = 2$, then, after 2 iterations of the while-loop, the degree is less than 3 and no more than 3 matchings are used at line 6, computing also a solution of value 3 or less. In both cases, $\lambda_{Gr}(G)$ is at most $2\chi'_S(G)$. In what follows, we assume that $\chi'_S(G) \geq 3$.
Note that $\lambda_{Gr}(G) \leq |\mathcal{S}'| + 1$ since $|\mathcal{S}'| \geq \Delta(G \setminus \mathcal{S}')$, where $G \setminus \mathcal{S}' = G[V \setminus \mathcal{S}']$. Let r be the last vertex introduced in $\mathcal{S}'$ and $\mathcal{S}'_* = \mathcal{S}' \setminus \{r\}$; we have $|\mathcal{S}'_*| < \Delta(G \setminus \mathcal{S}'_*)$ and consequently $d(r) \geq |\mathcal{S}'_*| + 1 = |\mathcal{S}'|$. Since vertices are introduced in $\mathcal{S}'$ in decreasing order of their

degree, every vertex in $\mathcal{S}'$ has degree at least $|\mathcal{S}'|$. Consequently, $|\{x : d(x) \geq |\mathcal{S}'|\}| \geq |\mathcal{S}'|$. It means that $|\mathcal{S}'| < \min\{d : |\{x : d(x) \geq d\}| < d\}$. It is straightforward to verify that $\min\{d : |\{x : d(x) \geq d\}| < d\} = k + 1$ and thus $\lambda_{Gr}(G) \leq |\mathcal{S}'| + 1 \leq k + 1$.
See Figure 6.1 for the illustration of the functions $y_1 = |\{x : d(x) > d\}|$ and $y_2 = |\{x : d(x) \geq d\}|$; then, k can be easily computed on the graphic as being the smallest value of d such that the function $y_1 = |\{x : d(x) > d\}|$ remains under the "graph" of the identity function (or coincides with it).

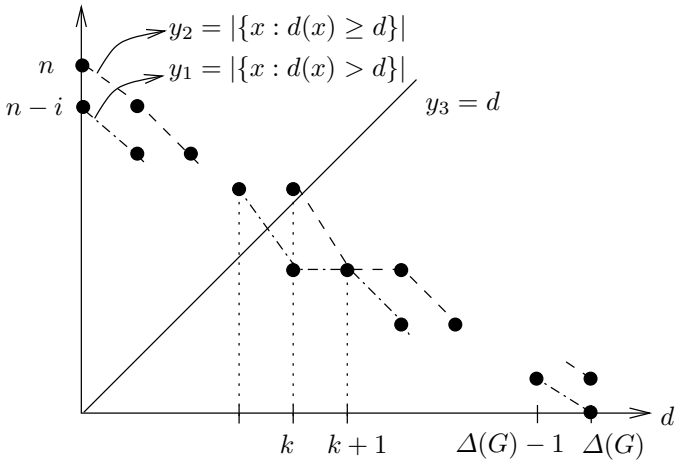


Figure 6.1: Computation of $k = \min\{d : |\{x : d(x) > d\}| \leq d\}$ in a graph G.

In order to show $k \leq 2\chi'_S(G)$, we prove the following lemma.

Lemma 6.12. *Consider an optimal edge split-coloring of value $\chi'_S(G)$ minimizing the number of triangles among optimal edge split-colorings of G. Denote by T the set of triangles and by B the set of bundles in this solution ($|T| + |B| \leq \chi'_S(G)$). Let X be the set of vertices of degree at least $2\chi'_S(G) + 1$ that are not center of a bundle in B. Then $|X| \leq 3$.*

Proof. Let $x \in X$, we denote by T_x the set of triangles in T incident to x and by B_x the set of bundles centered on neighbors of x (by definition of X, x is not a center of a bundle in B). Since the solution minimizes the number of triangles, any two triangles in T are edge-disjoint and no center of a bundle in B belongs to a triangle in T. Consequently, $T_x \cup B_x$ contains exactly $|B_x| + 2|T_x|$ edges incident to x. Since only $\chi'_S(G)$ edges incident to x can be covered by matchings in the solution, $|B_x| + 2|T_x| \geq \chi'_S(G) + 1$. Let us then define a bipartite graph $I = (X, T \cup B, E_I)$, with $xr \in E_I$ if and only if $r \in T_x \cup B_x$, i.e., x is incident to an edge of bundle or triangle r. Vertices in T have a degree at most 3 in I and vertices in B have a degree at most $|X|$ in I. We then have:

$$\sum_{x \in X} (|B_x| + 2|T_x|) \geq (\chi'_S(G) + 1)|X|, \tag{6.1}$$

$$\sum_{x \in X} (|B_x| + |T_x|) \leq 3|T| + |X||B|. \tag{6.2}$$

We deduce by subtraction:

$$3|T| \geq \sum_{x \in X} |T_x| \geq (\chi'_S(G) - |B| + 1)|X| - 3|T| \geq (|T| + 1)|X| - 3|T|.$$

Consequently $|X| \leq 5$. But, in this case, the number of triangles in T with degree 3 in I is at most two since any graph generated by three triangles and at most five vertices can be covered by three bundles, which is not possible if the solution minimizes the number of triangles. Then, if $|T| \geq 2$, inequality (6.2) can be replaced by $\sum_{x \in X}(|B_x| + |T_x|) \leq 2|T| + 2 + |X||B|$ implying $|X| \leq 4$. By the same argument as previously, since any graph generated by two triangles and at most four vertices can be covered by two bundles, at most one vertex in T has degree 3 in I implying $|X|(|T|+1) \leq 4|T|+2$ and thus $|X| \leq 3$. Finally if $|T| \leq 1$, inequality (6.2) becomes $\sum_{x \in X}(|B_x| + |T_x|) \leq 3 + |X||B|$ implying $|X| \leq 3$, which concludes the proof. □

It implies that $|\{x : d(x) > 2\chi'_S(G)\}| \leq \chi'_S(G) + 3 \leq 2\chi'_S(G)$ since $\chi'_S(G) \geq 3$. Then, $k \leq 2\chi'_S(G)$ and $\lambda_{Gr}(G) \leq 2\chi'_S(G) + 1$, which concludes the proof of *Statement 1.*
Statement 2. If $\chi'_S(G) \leq 2$, Algorithm 13 uses clearly no more than 3 colors. If $\chi'_S(G) \geq 3$, then by *Statement 1*, we have $\lambda_{Gr}(G) \leq 2\chi'_S(G) + 1 \leq 7\chi'_S(G)/3$.
Statement 3. Line-perfect graphs of maximum degree Δ can be edge colored in polynomial time (by $\mathcal{A}$) with Δ colors if $\Delta \geq 3$, and either with 2 or 3 colors if $\Delta = 2$. If $\Delta = 2$, Algorithm 13 uses at most 3 colors. If $\Delta \geq 3$, we just have to note that, in the proof of *Statement 1*, $\lambda_{Gr}(G) \leq k \leq 2\chi'_S(G)$. □

Let us finally remark that the bound is tight in bipartite graphs. Consider namely an integer p, $V_1 = \{x_i, i = 1, \ldots, 2p\}$, $V_2 = \{y_{ij}, i = 1, \ldots, 2p, j = 1, \ldots, p+1\} \cup \{u_i, i = 1, \ldots, p\}$, $E = \{x_i y_{ij}, i = 1, \ldots, 2p, j = 1, \ldots, p+1\} \cup \{x_i u_j, i = 1, \ldots, 2p, j = 1, \ldots, p\}$. Every vertex in V_1 is of degree $2p + 1 = \Delta(B)$, $d(u_i) = 2p, i = 1, \ldots, p$ and vertices $y_{ij}, i = 1, \ldots, 2p, j = 1, \ldots, p+1$ are of degree 1. The greedy algorithm removes vertices in V_1 (the related value being $2p$) while the optimal value $p + 1$ is achieved by removing $u_1, \ldots, u_p$. The related ratio is $2 - 2/(p + 1)$ and consequently the bound is asymptotically tight. The bound 2 is achieved for the same instance without vertices $y_{i(p+1)}$; in this case, if the greedy algorithm makes the bad choices, it may compute a solution of value $2p$ only.

Proposition 6.13. *For any graph G, Algorithm 14 gives a polynomial time 2-approximation for Min Edge Cocoloring.*

Proof. Let us consider a minimum cocoloring minimizing the number of triangles. Then it is straightforward to verify that it contains either two disjoint triangles, or one, or none (since all other solutions can be replaced by solutions of the same value and containing less triangles).
Algorithm 14 computes in polynomial time k minimizing $k+1+|\{x : d(x) > k\}|$; then there

Algorithm 14 Greedy edge cocoloring

Input: a graph G, edge coloring algorithm $\mathcal{A}$

Output: an edge cocoloring $\mathcal{Z}'$ of G

1. $\mathcal{Z}' \leftarrow \emptyset$;
2. **while** $\Delta(G \setminus M) + |M| < \Delta(G)$ **do** $//M = \{x : d(x) = \Delta(G)\}$
3. $\quad \mathcal{Z}' \leftarrow \mathcal{Z}' \cup M$;
4. $\quad G \leftarrow G[V \setminus M]$;
5. compute an edge coloring of the remaining edges by $\mathcal{A}$.

is an edge cocoloring consisting in bundles $\{x : d(x) > k\}$ (represented by their central vertices) completed by (at most) $k+1$ matchings. Hence, we can construct such a solution with $k + 1 + |\{x : d(x) > k\}|$ color classes.

Let us first suppose that the fixed minimum edge cocoloring does not contain any triangle. Then, $|\{x : d(x) > z'(G)\}| \leq z'(G)$ since all bundles of size greater than $z'(G)$ have to be taken as bundles in an optimal solution. Moreover, if $|\{x : d(x) > z'(G)\}| = z'(G)$, an optimal solution (containing only bundles) has been detected at a stage of the computation of k. So we can assume $|\{x : d(x) > z'(G)\}| \leq z'(G) - 1$, but in this case, by definition of k we have:

$$k + 1 + |\{x : d(x) > k\}| \leq z'(G) + 1 + z'(G) - 1 = 2z'(G).$$

If the optimal solution contains some triangles (one or two), one can consider all possible triangles in a solution and then apply the previous argument to the remaining graph. This completes the proof showing that one can compute a 2-approximation of Min Edge Cocoloring in polynomial time. □

Let us now consider Min Edge Split-coloring in bipartite graphs. Given a bipartite graph $B = (V_1, V_2, E)$ and an integer k, let us denote by $d'^k(x) = |N(x) \cap \{y : d(y) \leq k\}|$ the degree of x in the graph obtained by removing all neighbors of x of degree greater than k. For $i = 1, 2$, we also denote by $V_i^k = \{x \in V_i : d(x) > k\}$ and by $V'^{k,k'}_i = \{x \in V_i : d'^k(x) > k'\}$. For instance, $V'^{k,k'}_2$ is the set of vertices in V_2 with a degree greater than k' in the graph obtained by deleting all vertices of V_1 of degree greater than k. Finally, for $i \in \{1, 2\}$, we set $\bar{i} = 3 - i$, i.e., $\{1, 2\} = \{i, \bar{i}\}$.

In what follows, we derive an approximation algorithm for Min Edge Split-coloring in bipartite graphs. It computes three different edge split-colorings and returns the best one among them. The first one is obtained by removing V_1^k and V_2^k such that $|V_1^k| + |V_2^k| \leq k$ and that k is minimum. The second one consists of removing first $V_1^{\frac{1+\epsilon}{2-\epsilon}k}$ (vertices of V_1 of degree strictly greater than $\frac{1+\epsilon}{2-\epsilon}k$) and then $V'^{\frac{1+\epsilon}{2-\epsilon}k,k}_2$ (vertices in V_2 with degree greater than k in the graph $G[V \setminus V_1^{\frac{1+\epsilon}{2-\epsilon}k}]$) such that $|V_1^{\frac{1+\epsilon}{2-\epsilon}k}| + |V'^{\frac{1+\epsilon}{2-\epsilon}k,k}_2| \leq k$ and that k is minimum. The last solution is then calculated by interchanging the roles of V_1 and V_2.

Algorithm 15 Approximation of Min Edge Split-coloring of bipartite graphs

Input: a bipartite graph $B = (V_1, V_2, E)$, optimal edge coloring algorithm $\mathcal{A}$ in bipartite graphs (in time $\mathcal{O}(\Delta m)$)

Output: an edge split-coloring of B

1. $\epsilon \leftarrow (5 - \sqrt{17})/4$;
2. **for** $i = 1, 2$ **do**
3. for every $x \in V_i$ compute $d(x)$;
4. for every $y \in V_{\bar{i}}$ and every $x \in N(y)$, compute $d'^{d(x)}(y)$;
5. for every $k \in \{1, \ldots, \Delta(B)\}$ compute $|V_i^k|$ and $|V'^{\frac{1+\epsilon}{2-\epsilon}k,k}_{\bar{i}}|$;
6. $d_i \leftarrow \min\{k : |V_i^{\frac{1+\epsilon}{2-\epsilon}k}| + |V'^{\frac{1+\epsilon}{2-\epsilon}k,k}_{\bar{i}}| \leq k\}; \mathcal{S}'_i \leftarrow V_i^{\frac{1+\epsilon}{2-\epsilon}d_i} \cup V'^{\frac{1+\epsilon}{2-\epsilon}d_i,d_i}_{\bar{i}}$;
7. $d_0 \leftarrow \min\{k : |V_1^k| + |V_2^k| \leq k\}; \mathcal{S}'_0 \leftarrow V_1^{d_0} \cup V_2^{d_0}$;
8. $i_0 \leftarrow \text{argmin}\{d_i, i = 0, 1, 2\}; \mathcal{S}' \leftarrow \mathcal{S}'_{i_0}$;
9. compute an edge coloring of the remaining edges by $\mathcal{A}$.

Theorem 6.14. *Algorithm 15 is a $\mathcal{O}(mn)$ time algorithm approximating Min Edge Split-coloring in bipartite graphs within ratio $2 - (5 - \sqrt{17})/4 \simeq 1.78$.*

Proof. We take $\epsilon = (5 - \sqrt{17})/4 \simeq 0.22$ as defined in the algorithm. It is the root of $1+\epsilon = 2(1-\epsilon)^2$ which is smaller than 1. It follows that $2-\epsilon \simeq 1.78$ and $(1+\epsilon)/(2-\epsilon) \simeq 0.68$. Let us first note that $d = d_{i_0} = \min\{d_0, d_1, d_2\}$, where $d_0 = \min\{k : |V_1^k \cup V_2^k| \leq k\}$ and for $i = 1, 2$, $d_i = \min\{k : |V_i^{\frac{1+\epsilon}{2-\epsilon}k}| + |V'^{\frac{1+\epsilon}{2-\epsilon}k,k}_{\bar{i}}| \leq k\}$. Moreover, it is immediate to verify that Algorithm 15 computes a feasible edge split-coloring of value d. More precisely, d is such that the graph obtained by removing at most d vertices is of degree at most d: the maximum degree of the graph obtained by removing $V_1^{d_0} \cup V_2^{d_0}$ is at most d_0 and the graph obtained by removing $V_i^{\frac{1+\epsilon}{2-\epsilon}d_i} \cup V'^{\frac{1+\epsilon}{2-\epsilon}d_i,d_i}_{\bar{i}}$ has degrees at most d_i, $i = 1, 2$ (note that $(1+\epsilon)/(2-\epsilon) \leq 1$). Concerning the complexity, lines 3, 5, 6 and 7 need $\mathcal{O}(m)$ steps while line 4 needs $\mathcal{O}(mn)$. Furthermore, the optimal edge coloring algorithm $\mathcal{A}$ given in [108] coloring the edges of a bipartite graph with Δ colors runs in time $\mathcal{O}(\Delta m)$ which is dominated by $\mathcal{O}(mn)$; hence the overall complexity is $\mathcal{O}(mn)$.

Let us now analyze the approximation behavior of the algorithm. Denote by $\ell = \chi'_S(B)$: $\exists L_1 \subset V_1, L_2 \subset V_2, |L_1| = \ell_1, |L_2| = \ell_2, \ell_1 + \ell_2 = \ell$ and $\Delta(B \setminus (L_1 \cup L_2)) \leq \ell$, where $B \setminus (L_1 \cup L_2) = B[(V_1 \cup V_2) \setminus (L_1 \cup L_2)]$. In the sequel, we consider the following cases.

Case 1. $\ell_1 \geq \ell\epsilon$ and $\ell_2 \geq \ell\epsilon$.

Case 2. $\ell_2 < \ell\epsilon$ with the following two sub-cases.

Sub-case 2.1. $|V_1^{\ell(1+\epsilon)}| \geq \ell\epsilon$.

Sub-case 2.2. $|V_1^{\ell(1+\epsilon)}| < \ell\epsilon$.

Case 3. $\ell_1 < \ell\epsilon$.

Let us point out the following property **(P)** which will be useful.

(P) If $x \in V_i \setminus L_i, i \in \{1,2\}$ and $d(x) \geq \ell + r$, then $|N(x) \cap L_{\bar{i}}| \geq r$ where $\bar{i} = 3 - i$.

This holds because after removal of $L_{\bar{i}}$, vertex x has degree at most ℓ.

Case 1. $\ell_1 \geq \ell\epsilon$ and $\ell_2 \geq \ell\epsilon$.

By Property **(P)**, $V_i^{\ell+l_{\bar{i}}} \subset L_i$ for $i = 1, 2$, then:

$$|V_1^{\ell+\max(\ell_1,\ell_2)} \cup V_2^{\ell+\max(\ell_1,\ell_2)}| \leq |V_1^{\ell+\ell_2}| + |V_2^{\ell+\ell_1}| \leq \ell_1 + \ell_2 = \ell \leq \ell + \max(\ell_1, \ell_2).$$

We deduce, $d_0 \leq \ell + \max(\ell_1, \ell_2) \leq \ell(2-\epsilon)$ where the last inequality holds since we are considering Case 1.

Case 2. $\ell_2 < \ell\epsilon$.

By Property **(P)**, we have $V_1^{\ell(1+\epsilon)} \subset L_1$.

Sub-case 2.1. $|V_1^{\ell(1+\epsilon)}| \geq \ell\epsilon \Rightarrow |(L_1 \setminus V_1^{\ell(1+\epsilon)})| \leq \ell(1-\epsilon)$.

Then, Property **(P)** implies that $V'^{\ell(1+\epsilon),\ell(2-\epsilon)}_2 \subseteq L_2$. It follows from the above relations that $|V_1^{\ell(1+\epsilon)} \cup V'^{\ell(1+\epsilon),\ell(2-\epsilon)}_2| \leq \ell \leq (2-\epsilon)\ell$, which implies, by definition of d_1 (consider $k = \ell(2-\epsilon)$ in the definition), that $d_1 \leq \ell(2-\epsilon)$.

Sub-case 2.2. $|V_1^{\ell(1+\epsilon)}| < \ell\epsilon$.

For every $x \in V_2 \setminus L_2$ such that $d'^{\ell(1+\epsilon)}(x) > \ell(2-\epsilon)$, we have by Property **(P)** $|N(x) \cap (L_1 \setminus V_1^{\ell(1+\epsilon)})| \geq \ell(1-\epsilon)$. Then, by considering the number $\mathcal{E}$ of edges between $(L_1 \setminus V_1^{\ell(1+\epsilon)})$ and $(V'^{\ell(1+\epsilon),\ell(2-\epsilon)}_2 \setminus L_2)$, we deduce:

$$(|V'^{\ell(1+\epsilon),\ell(2-\epsilon)}_2| - \ell_2)\ell(1-\epsilon) \leq \mathcal{E} \leq (\ell_1 - |V_1^{\ell(1+\epsilon)}|)\ell(1+\epsilon) \leq \ell_1\ell(1+\epsilon)$$

since the maximum degree of V_1 after removing $V_1^{\ell(1+\epsilon)}$ is at most $\ell(1+\epsilon)$.

It follows that:

$$|V'^{\ell(1+\epsilon),\ell(2-\epsilon)}_2| \leq \frac{\ell(1+\epsilon)}{1-\epsilon} = \ell(2-2\epsilon).$$

Consequently $|V_1^{\ell(1+\epsilon)}| + |V'^{\ell(1+\epsilon),\ell(2-\epsilon)}_2| \leq \ell(2-\epsilon)$, which implies $d_1 \leq \ell(2-\epsilon)$.

Case 3. $\ell_1 < \ell\epsilon$.

It corresponds to the second case by interchanging V_1 and V_2. Therefore $d_2 \leq \ell(2-\epsilon)$ and in all cases, $d = \min\{d_0, d_1, d_2\}$ satisfies the expected ratio.

□

6.4 Comparability graphs

Comparability graphs are defined as graphs admitting a transitive orientation of their edges; i.e., an orientation such that if $ab \in E$ is oriented from a to b and $bc \in E$ from b to c then necessarily $ac \in E$ and a is oriented towards c. They are a subclass of perfect graphs [80]. Let us first note the following result allowing us to deduce the hardness of Min Split-coloring in comparability graphs.

Theorem 6.15. *Let $\mathcal{G}$ be a class of graphs closed under addition of disjoint cliques without link to the rest of the graph and under addition of a complete k-partite graph completely linked to the rest of the graph. Then, Min Cocoloring reduces in polynomial time to Min Split-coloring in the class $\mathcal{G}$.*

Proof. Let us consider a graph G of size n such that $z(G) = p + k$ (where p is the number of cliques and k is the number of stable sets in an optimum solution) and let us first assume that $p \leq k$. Consider the graph G' consisting of G and $l = k - p \leq n$ disjoint cliques, each of size $n + 1$, without any link to the rest of the graph. Note that $k - p$ new cliques completed by p cliques and k stable sets of the optimal cocoloring of G form a split-coloring of value k, implying that $\chi_S(G') \leq k \leq n$. Consequently a minimum split-coloring of G' necessarily contains the $k - p$ new cliques completed by p' cliques and k' stable sets of G. Since $\chi_S(G') = \max((k - p + p'), k') \leq k$, we have $p' \leq p$ and $k' \leq k$. On the other hand, $p' + k' \geq k + p$ since the restriction to G of the split coloring of G' provides a cocoloring of value $p' + k'$. Thus $p' + k' = p + k$ and this cocoloring of G is optimal.
If $z(G) = p + k$ with $p \geq k$, we show by the same arguments that a minimum cocoloring of G can immediately be deduced from a minimum split coloring of G'', the graph obtained from G by adding $p - k \leq n$ stable sets, each of size $n + 1$ and completely linked to the rest of the graph.
Finally, in both cases, $|k - p| \leq k + p \leq 2\chi_S(G)$, consequently, the reduction runs as in Algorithm 16.

Algorithm 16 Polynomial reduction of Min Cocoloring to Min Split-coloring

Input: a graph G as an instance of Min Cocoloring, an exact algorithm for Min Split-coloring

Output: an optimal cocoloring of G

1. $\mathcal{P} \leftarrow \emptyset$; *//$\mathcal{P}$ will contain cocolorings of G*;
2. compute an optimal split-coloring of G;
3. store in $\mathcal{P}$ the related partition; $L \leftarrow 2\chi_S(G)$;
4. **for all** $l \in \{1, \ldots, L\}$ **do**
5. construct $G' = G \cup lK_{n+1}$;
6. compute an optimal split-coloring of G' and store its restriction to G in $\mathcal{P}$;
7. construct $G'' = K \oplus \overline{lK_{n+1}}$;
8. compute an optimal split-coloring of G'' and store its restriction to G in $\mathcal{P}$;
9. **return** the best cocoloring stored in $\mathcal{P}$.

□

Note that the class $\mathcal{G}$ just described is a strict superclass of perfect graphs; it implies among others the following corollary that will guide us in our search for a class of graphs where Min Split-coloring is polynomially solvable and Min Cocoloring is $\mathcal{NP}$-hard.

Corollary 6.16. *There is no subclass of perfect graphs where Min Split-coloring is polynomially solvable while Min Cocoloring is $\mathcal{NP}$-hard.*

Let us, for a while, consider a subclass of comparability graphs, called permutation graphs; a graph G is a *permutation graph* if G and $\overline{G}$ are comparability graphs. Another immediate implication of Theorem 6.15 follows from the fact that Min Cocoloring is $\mathcal{NP}$-hard in permutation graphs [117].

Corollary 6.17. *Min Split-coloring is $\mathcal{NP}$-hard in permutation graphs.*

Having established the $\mathcal{NP}$-hardness of Min Split-coloring in comparability graphs, we show that the method proposed in [65], for approximating Min Cocoloring in comparability graphs within a factor of 1.71, can be adapted to Min Split-coloring. Note that a graph G is a *cocomparability graph* if $\overline{G}$ is a comparability graph.

Theorem 6.18. *Algorithm 18 gives a 2-approximation of Min Split-coloring for comparability and cocomparability graphs in time $\mathcal{O}(n^{7/2} + n^{3/2}m)$.*

Proof. Let us first establish the split counterpart of Lemma 2 in [65].

Lemma 6.19. *Let $G = (V, E)$ be a perfect graph of order n and let k satisfy $k \geq \sqrt{n}$, then $\chi_S(G) \leq k$ and a split-coloring of size k can be computed in polynomial time.*

Proof. Let $G = (V, E)$ be a perfect graph, in Procedure 17, we consider a slight modification of Procedure SQRTPartition of [65].

Procedure 17 SQRT-Split-partition

Input: a perfect graph G, an integer k such that $k \geq \sqrt{n}$

Output: a k-split-coloring of G

1. **while** $k \neq 0$ and G is not empty **do**
2. **if** $\min(\alpha(G), \alpha(\overline{G})) \leq k$ **then**
3. compute a k-coloring of G or $\overline{G}$, include each clique or stable set in the solution and set $k \leftarrow 0$;
4. **else**
5. find a stable set and a clique of size $k+1$ and color the related split graph of size at least $2k+1$ with a new color;
6. set $k \leftarrow k-1$ and remove from G all already colored vertices.

It is straightforward to verify that Procedure 17 runs in polynomial time. Moreover, if line 3 is executed or if the graph becomes empty, then it computes a split-coloring of size k. If line 3 is not computed and if k loops are performed, then at least $\sum_{i=0}^{k-1} 2(k-i)+1 = k(k+2) \geq k^2$ vertices are covered and consequently the graph is also covered by k split graphs. □

Now, let us analyze Algorithm 18 which is an adaptation of the algorithm `APPROX COCOLOURING` of [65] for Min Split-coloring.

Algorithm 18 Approximation of Min Split-coloring of comparability graphs

Input: a comparability graph G, Procedure 17

Output: a split-coloring of G

1. compute a maximum r-colorable subgraph (C_r, E_r) of $\overline{G}$ and a maximum r-colorable subgraph (S_r, E'_r) of G such that r is minimum subject to $|C_r| + |S_r| \geq n$;
2. introduce in the solution an r-split-coloring of $C_r \cup S_r$;
3. remove $C_r \cup S_r$ from G;
4. complete the solution by the split graphs computed by Procedure 17 in the remaining graph.

Since G can be decomposed into $\chi_S(G)$ cliques and $\chi_S(G)$ stable sets, we have $r \leq \chi_S(G)$, where r is as described in Algorithm 18. On the other hand, since $|C_r \cap S_r| \leq r^2$, we have $n - |C_r \cup S_r| \leq r^2$ and consequently, by Lemma 6.19, at most $r \leq \chi_S(G)$ split graphs are computed at line 4. The computed split-coloring is of size at most $2\chi_S(G)$ and the proof is complete. Note that this result remains valid for every class of perfect graphs for which subgraphs such as described in line 1 of Algorithm 18 can be polynomially computed.
The complexity of steps 1, 2 and 3 is $\mathcal{O}(\chi_S(G)n^3) \leq \mathcal{O}(n^{7/2})$; it follows from the fact that a maximum r-colorable subgraph of G and $\overline{G}$ can be computed in time $\mathcal{O}(n^3)$ in comparability graphs [74] and that $\chi_S(G) \leq \sqrt{n}$. Let us now analyze the complexity of Procedure 17 for comparability and cocomparability graphs. Line 5 of Procedure 17 is computed at most t times, where t is the smallest integer such that $(2k+1)+(2(k-1)+1)+\ldots+(2(k+1-t)+1) \geq n$ or equivalently $t^2 - (2k+2)t + n \leq 0$; hence, recalling that we have $k \geq \sqrt{n}$,

$$\begin{aligned} t &= \left\lceil \frac{(2k+2) - \sqrt{(2k+2)^2 - 4n}}{2} \right\rceil \\ &\leq \left\lceil \frac{4n}{2\Big((2k+2) + \sqrt{(2k+2)^2 - 4n}\Big)} \right\rceil \\ &\leq \left\lceil \sqrt{n} \right\rceil \\ &\leq \sqrt{n} + 1. \end{aligned}$$

Finding a maximum clique and a maximum stable set in a comparability graph can be done respectively in time $\mathcal{O}(n+m)$ and $\mathcal{O}(nm)$; therefore, the complexity of this step is dominated by $\mathcal{O}(n^{3/2}m)$. This completes the proof of the overall complexity. □

6.5 General graphs

6.5.1 Standard approximation

Min Coloring is known to be particularly difficult to approximate since it is not approximable within $n^{1-\epsilon}$ if co$\mathcal{RP}$[5]$\neq \mathcal{NP}$ and not approximable within $n^{1/7-\epsilon}$ if $\mathcal{P}\neq\mathcal{NP}$ [7]. Similar hardness results can immediately be deduced for Min Split-coloring and Min Cocoloring.

Proposition 6.20. *The following statements hold.*

1. *If Min Cocoloring is $n^{1/2-\epsilon}$-approximable for $0 < \epsilon < 1/2$, then Min Coloring is $n^{1-\epsilon}$-approximable.*
2. *If co$\mathcal{RP}$ $\neq\mathcal{NP}$, then for every $\epsilon > 0$, Min Cocoloring is not approximable within $n^{1/2-\epsilon}$. If $\mathcal{P}\neq\mathcal{NP}$, then for every $\epsilon > 0$, Min Cocoloring is not approximable within $n^{1/14-\epsilon}$.*
3. *Statements 1 and 2 hold for Min Split-coloring.*

Proof. Let $\mathcal{O}$ be an oracle for Min Cocoloring guaranteeing the ratio $n^{1/2-\epsilon}$, with $\epsilon < 1/2$. The reduction constructs $\tilde{G}$ consisting in $\lfloor n^{1-\epsilon}\rfloor + 1$ copies of G without link and computes a cocoloring of $\tilde{G}$ by using $\mathcal{O}$. If a copy of G in $\tilde{G}$ is covered only by stable sets, then it outputs this coloring; else it outputs any greedy coloring.
If $\chi(G) \leq n^{\epsilon}$, then $z(\tilde{G}) \leq \chi(\tilde{G}) = \chi(G) \leq n^{\epsilon}$. As the cocoloring computed by the oracle on $\tilde{G}$ guarantees the ratio $n(\tilde{G})^{1/2-\epsilon}$ and $n(\tilde{G}) \leq n^2$, it uses at most $(n^2)^{1/2-\epsilon}n^{\epsilon} = n^{1-\epsilon}$ colors. Consequently, at least one copy of G in $\tilde{G}$ is covered only by stable sets in the cocoloring computed by $\mathcal{O}$, which leads to a coloring of G using at most $n^{1-\epsilon}$ colors and the ratio $n^{1-\epsilon}$ is guaranteed. If now $\chi(G) > n^{\epsilon}$, then any coloring of G uses at most n colors; this guarantees the expected ratio, which concludes the proof of *Statement 1*. *Statement 2* follows from hardness results for Min Coloring. Finally, *Statement 3* is immediately deduced by using the fact that for any graph G, we have $\chi_S(G) \leq z(G)$. □

This hardness result considerably limits the possibilities for approximating Min Split-coloring and Min Cocoloring in general graphs. A master-slave strategy enables us to reduce these problems to Max Stable Set and Max Clique with an increase of the ratios by a factor $\mathcal{O}(\log n)$ (see [4] and [109]). This is done by means of the greedy cocoloring algorithm given in Algorithm 8 (and its split-counterpart) where the "slave" problem (Max Stable Set and Max Clique) serves the "master" problem which is the Min Cocoloring (respectively Min

[5]The class $\mathcal{RP}$, short for *Randomized Polynomial Time*, is the class of languages for which membership can be determined in polynomial time by a probabilistic Turing machine with no false acceptance and less than half false rejections. It is known that $\mathcal{P} \subseteq \mathcal{RP} \subseteq \mathcal{NP}$. co$\mathcal{RP}$ is the class of problems such that their complementary (the problem with the reversed answers) belongs to $\mathcal{RP}$; that is, co$\mathcal{RP}$ consists in languages L that have a polynomial time randomized algorithm A erring only in the case when an instance x of a problem does not belong to L. See [106, 115] for more information on these complexity classes.

Split-coloring); at each step the vertices of a maximum stable set or (respectively, and) a maximum clique are covered. This trivially leads to a $\mathcal{O}(n/\log n)$-approximation for both problems; but it seems not so easy to reduce these problems to Min Coloring in order to refine the comparison of their approximation behavior.

6.5.2 Differential approximation

Theorem 6.21. *Algorithm 19 is a $\mathcal{O}(n^{3p+1})$ time algorithm guaranteeing a differential approximation ratio of $(1-1/p)$ for both Min Split-coloring and Min Cocoloring.*

Algorithm 19 DPTAS-split-coco

Input: a graph G, an integer p

Output: a split-coloring or a cocoloring of G with differential approximation ratio of $(1-1/p)$

1. **while** the current graph contains a $3p$-stable set or a $3p$-clique **do**
2. color such a stable set or clique with a new color;
3. complete the solution by an exhaustive search on the remaining graph.

Proof. For the whole complexity, note that step 3 is computed for a graph without a stable set or a clique of size $3p$, and consequently its size is less than the related Ramsey number $R_2(3p,3p) \leq K^p$ for a constant K [24].
It is straightforward to verify that Algorithm 19 computes either a split-coloring or a cocoloring of the instance. The only difference between the two cases arises in line 3 that computes either an optimal split-coloring or an optimal cocoloring in the remaining graph. We propose an analysis valid for both problems. The problem being fixed, for a given graph H, we respectively denote by $\omega(H)$ and $\beta(H)$ the worst value and the optimal value with respect to this problem (consequently $\beta(H)$ stands either for $\chi_S(H)$ or for $z(H)$).
The approximation ratio is proved by induction on $n(G)$ (see also [86]).
If $n(G) < 3p$, then only step 3 is computed and the algorithm finds an optimal solution corresponding to a ratio of 1. Let us now assume that the expected ratio is guaranteed for every graph of size n or less, where $n \geq 3p$, and consider a graph G_{n+1} of size $n+1$. If no clique or stable set of size $3p$ is detected at step 1, then G_{n+1} is optimally colored at step 3. Else, the algorithm attributes a new color either to a stable set or to a clique of size $3p$; it is then executed on the graph G' obtained from G_{n+1} by deleting these $3p$ vertices. Since G' is of size less than n, the ratio is guaranteed for G'. Note also that:

$$\begin{aligned} \lambda(G_{n+1}) &= 1+\lambda(G'), \\ \beta(G_{n+1}) &\geq \beta(G'), \\ \omega(G_{n+1}) &\geq \omega(G')+p \geq \lambda(G_{n+1}), \end{aligned}$$

which implies:

$$\begin{aligned}\omega(G') + p - \lambda(G_{n+1}) &\geq (1 - 1/p)(\omega(G') - \beta(G')) + p - 1 \\ &\geq (1 - 1/p)(\omega(G') + p - \beta(G_{n+1}))\end{aligned}$$

and then, since $\omega(G_{n+1}) \geq \omega(G')+p$ and δ is increasing with respect to ω (see Property 6.6), we have:

$$\frac{\omega(G_{n+1}) - \lambda(G_{n+1})}{\omega(G_{n+1}) - \beta(G_{n+1})} \geq \frac{\omega(G') + p - \lambda(G_{n+1})}{\omega(G') + p - \beta(G_{n+1})} \geq (1 - 1/p).$$

This concludes the proof. □

It is straightforward to verify that, since Min Split-coloring (respectively Min Cocoloring) has integral values and $\omega(G) - \chi_S(G)$ is polynomially bounded, a DFPTAS would allow to solve it polynomially for any finite graph.

6.6 Bounds on χ_S and z

Now, we terminate the study on the approximation behavior of generalized coloring problems. In this section, we develop some bounds on optimal split-colorings and cocolorings. The purpose is to obtain relatively good bounds with a limited effort in order to obtain "good" initial solutions for some possible heuristics.
First, we give some intuitive bounds for χ_S. In a second time, we extend the sequential algorithms for the usual vertex coloring due to Welsh-Powell [118] and Matula [101, 102] to the case of split-coloring and cocoloring.

6.6.1 Simple bounds

In the sequel, we first present a very intuitive method to obtain a lower bound on χ_S and then a more theoretical result for an upper bound on χ_S.

A lower bound for χ_S

Let us determine a procedure which gives an ordered set of induced maximal cliques in an arbitrary graph G.

Procedure 20 CliquesList

Input: a graph G

Output: a set $\mathcal{K}$ of disjoint (not linked in between) maximal cliques in G

1. **repeat**
2. choose a vertex v and find a maximal clique K^v containing v;
3. store K^v in $\mathcal{K}$, then remove K^v and its neighbors from G;
4. **until** $G = \emptyset$
5. reorder $\mathcal{K} = \{K^i\}$ in non-increasing order of clique sizes.

Procedure CliquesList returns a list $\mathcal{K}$ of induced maximal cliques in G. In other words, there is no edge linking vertices of any pair of cliques in $\mathcal{K}$. Moreover, this list is maximal in the sense that adding any new maximal clique in $\mathcal{K}$ introduces edges between cliques. An optimal split-coloring $\chi_S(\mathcal{K})$ of $\mathcal{K}$ constitutes a lower bound for $\chi_S(G)$ since, in the best case, $\chi_S(\mathcal{K})$ colors will be sufficient to color all the vertices of G; $\chi_S(\mathcal{K}) \leq \chi_S(G)$. Therefore, we will concentrate on the optimal split-coloring of a set of induced cliques of arbitrary sizes.

Our approach can be visualized in a diagram where cliques K^i are represented on the x axis as columns of length proportional to their cardinalities r_i. Ordering cliques in non-increasing order of their sizes implies that we can read, on the y axis, the number r_i^* of cliques K^k such that $r_k \geq i$. In this formulation, our problem consists in finding the largest k such that $\min(r_k, r_k^*) \geq k$. This amounts to finding the largest square that can be inserted under the stairs. We may use the following strategy: choose repetitively the largest remaining (not entirely colored) clique as the clique of a new split graph G_i (or color) and one vertex (not colored) from every other clique as the stable set of the same split graph. Each G_i is represented in Figure 6.2 by a grey broken line (with breakpoint at entry (i,i)). Repeating this until no vertex remains uncolored gives rise to a split-coloring which uses exactly k colors. In Figure 6.2 we have $\mathcal{K} = \{K_8^1,\ K_8^2,\ K_7^3,\ K_6^4,\ K_6^5,\ K_5^6,\ K_3^7,\ K_3^8,\ K_2^9,\ K_2^{10},\ K_1^{11}\}$ and we obtain a 5-split-coloring, which is optimal.

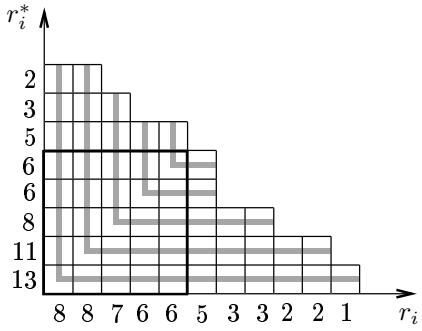


Figure 6.2: Optimal split-coloring of $\mathcal{K}$.

An upper bound for χ_S

Assume that we have a k-coloring (not necessarily an optimal coloring) of G given by stable sets $(S^1, \ldots, S^k)$. For any $p \leq k$, an optimal clique cover of $(S^1 \cup \ldots \cup S^p)$ together with an optimal coloring of $(S^{p+1} \cup \ldots \cup S^k)$ constitutes a split-coloring of G in $\max\left\{\theta(S^1 \cup \ldots \cup S^p), \chi(S^{p+1} \cup \ldots \cup S^k)\right\}$ colors. Therefore, the split-chromatic number of G, $\chi_S(G)$, would be less than or equal to the minimum on p of this quantity. Furthermore, one can determine the minimum on any possible p-tuple of stable sets. Hence, the upper bound is expressed in the following way:

$$\chi_S(G) \leq \min_p \left\{ \min_{S^{i^1} \cup \ldots \cup S^{i^p}} \left\{ \max\{\theta(S^{i^1} \cup \ldots \cup S^{i^p}), \chi(G - (S^{i^1} \cup \ldots \cup S^{i^p}))\}\right\}\right\}.$$

Beyond the theoretical interest of this upper bound, one can identify cases where it can be efficiently used in practice. First of all, if we can compute $\chi(G)$ in polynomial time then we will be able to take into account a smaller number of stable sets while searching for the minimum on p. For instance, this is the case for perfect graphs. Furthermore, in this context, $\theta(S^{i^1} \cup \ldots \cup S^{i^p})$ and $\chi(G - (S^{i^1} \cup \ldots \cup S^{i^p}))$ can also be computed in polynomial time, providing an overall complexity which is polynomial, to obtain an upper bound. One may search for conditions (on G or on the optimal coloring of G) for this upper bound to be tight. Note that, once more, we are more likely to find such a condition on perfect graphs.

6.6.2 Welsh-Powell sequential algorithm

Let us first concentrate on the split-coloring problem. Bounds on other (p,k)-coloring problems will be then easily derived by the same approach.
By analogy with the well known sequential coloring algorithm of Welsh and Powell [118], we may devise a similar procedure for the split-coloring problem.
We start by ordering the vertices $v_1, v_2, \ldots, v_n$ of a graph $G = (V, E)$ in non-decreasing order of degrees, i.e., $d(v_1) \leq d(v_2) \leq \ldots \leq d(v_n)$. Then we color sequentially the vertices of G starting from v_1 and using the smallest available color; thus the color $c(v_i)$ given to vertex v_i will satisfy $c(v_i) \leq \min\{i, d(v_i) + 1\}$. This partial coloring will give us the stable sets of the different split graphs to be determined for obtaining a split-coloring of G.
In addition to this, we start from v_n and we construct the cliques which are in the different split graphs. For doing so, we may consider the complement $\overline{G}$ of G in which the degrees $\bar{d}(v_i)$ are given by $n - 1 - d(v_i)$; thus we have $\bar{d}(v_n) \leq \bar{d}(v_{n-1}) \leq \ldots \leq \bar{d}(v_1)$. Constructing the cliques in G is equivalent to constructing a coloring (in the usual sense) of $\overline{G}$. Therefore we start from v_n and apply the sequential coloring algorithm; the color $\bar{c}(v_i)$ of vertex v_i will satisfy $\bar{c}(v_i) \leq \min\{n - i + 1, \bar{d}(v_i) + 1\} = \min\{n - i + 1, n - d(v_i)\}$ for $i = n, n-1, \ldots, 1$. Starting from both ends, we get to a point where we have two consecutive vertices v_l and v_{l+1} such that v_l has been colored in G by starting from v_1, and v_{l+1} in $\overline{G}$ by starting from v_n. We compute

$$A_{WP}(l) = \max\big\{\min\{l, d(v_l) + 1\}, \min\{n - l, n - d(v_{l+1})\}\big\}.$$

Then, vertices of G can be covered by $A_{WP}(l)$ split graphs; they are obtained by taking in S_1 the vertices v_i of G with color 1 and with $i \leq l$; in K_1 we take the vertices v_i of G with color 1 and $i \geq l + 1$. This gives the first split graph in the split-coloring. We continue in the same way for colors $2, 3, \ldots, A_{WP}(l)$. Now we have to find

$$\chi_S^{WP}(G) = \min_{1 \leq l \leq n-1} A_{WP}(l)$$

in order to determine at which vertex v_l we have to stop in coloring G from v_1 and $\overline{G}$ from v_n. So we obtain the bound of Proposition 6.22 for the split-chromatic number $\chi_S(G)$.
On the other hand, it is easily seen that replacing $A_{WP}(l)$ by

$$Z_{WP}(l) = \min\{l, d(v_l) + 1\} + \min\{n - l, n - d(v_{l+1})\}$$

and taking the minimum on l gives a bound on the cocoloring problem. As for the more general (p,k)-coloring problems, the vertex where we have to stop when coloring G from v_1 and $\overline{G}$ from v_n is decided during the algorithm; in the (p,k_{min})-coloring (respectively (p_{min},k)-coloring) problem, we stop at the first vertex l which can not be colored by a color $c(l) \leq p$ (respectively $c(l) \leq k$) in coloring G from v_1 (respectively $\overline{G}$ from v_n) and the remaining vertices are colored starting from v_n (respectively v_1).

Proposition 6.22. *For any graph G, we have $\chi_S(G) \leq \bar{\chi}_S^{WP}(G) = \min_{1\leq l\leq n-1} A_{WP}(l)$ and $z(G) \leq \bar{z}^{WP} = \min_{1\leq l\leq n-1} Z_{WP}(l)$.*

6.6.3 Matula sequential algorithm

One can exploit the analogy of the above coloring procedure with the procedure "smallest last" (SL) of Matula [101, 102], where one places at the last available position of the order a vertex with a minimum degree in the remaining graph. Such an ordering (smallest last) gives a bound of $1 + \max\{\min\{d_H(v) : v \in H\} : H \text{ subgraph of } G\}$ for the chromatic number. The difficulty is that this order may not be the same as the one obtained by inserting, at the beginning of the order, a vertex with largest degree in the remaining graph. To avoid this problem, we may fix a priori the vertices to be colored as cliques and the vertices that will be in some stable sets. To do so, one solution consists in splitting the graph G into two graphs: $G_S = G[\{v_1, v_2, \ldots, v_l\}]$, where $v_1, v_2, \ldots, v_l$ are l vertices of smallest degrees in G, and $G_K = G[\{v_{l+1}, v_{l+2}, \ldots, v_n\}]$, where $v_{l+1}, v_{l+2}, \ldots, v_n$ are the remaining vertices, i.e., the $n-l$ vertices of largest degrees in G. Now, one can apply the SL algorithm to G_S for obtaining stable sets and to $\overline{G}_K$ for obtaining cliques. Obviously, we use at most $\max_{H\subseteq G_S}\{1 + \min_{v\in H}\{d_H(v)\}\}$ stable sets and at most $\max_{\bar{H}\subseteq G_K}\{1 + \min_{v\in \bar{H}}\{d_{\bar{H}}(v)\}\}$ cliques. Let us denote

$$A_M(l) = \max\Big\{ \max_{H\subseteq G_S} \big\{1 + \min_{v\in H}\{d_H(v)\}\big\}, \max_{\bar{H}\subseteq G_K} \big\{1 + \min_{v\in \bar{H}}\{d_{\bar{H}}(v)\}\big\}\Big\},$$

$$Z_M(l) = \max_{H\subseteq G_S} \big\{1 + \min_{v\in H}\{d_H(v)\}\big\} + \max_{\bar{H}\subseteq G_K} \big\{1 + \min_{v\in \bar{H}}\{d_{\bar{H}}(v)\}\big\},$$

then, the following bounds are obtained.

Proposition 6.23. *For any graph G, we have $\chi_S(G) \leq \bar{\chi}_S^{M}(G) = \min_{1\leq l\leq n-1} A_M(l)$ and $z(G) \leq \bar{z}^{M} = \min_{1\leq l\leq n-1} Z_M(l)$.*

Once again, the above version of SL algorithm may be used to obtain some bounds on general (p,k)-coloring problems. Different methods can be developed to decide at which vertex the algorithm stops. For instance, one may try to assign to p cliques as many as possible of the vertices of largest degrees in G. This can be done by taking the maximum number n_p, such that applying the SL algorithm on the complement of the graph induced by n_p vertices of largest degrees, necessitates not more than p colors. Then, a bound on k is obtained by applying the SL algorithm to the remaining graph.
Both of the above approaches are built on the idea of adapting Welsh-Powell and SL (Matula)

algorithms in such a way that vertices of small degrees are assigned to stable sets and vertices of large degrees are in cliques, which is intuitively correct. This approach is certainly more appropriate to (p,k)-coloring problems even though its positive effect is not directly observed on the theoretical bound obtained.
As for the complexities of these algorithms, both the computations of $A_{WP}(l)$ and $A_M(l)$ (as well as $Z_{WP}(l)$ and $Z_M(l)$) are made by one search of vertices and therefore is sequential. In return, taking the minimum on every possible values of l prevents the procedure from remaining sequential.
It is easy to observe that the SL algorithm of Matula gives a bound on the chromatic number which is less than or equal to the Welsh-Powell bound. Since this inequality holds for both of the terms of A_M and A_{WP} (and also Z_M and Z_{WP}), one may conclude that for any graph G, we have $\bar{\chi}_S^M(G) \leq \bar{\chi}_S^{WP}(G)$ and also $\bar{z}_S^M(G) \leq \bar{z}_S^{WP}(G)$, i.e., the theoretical bounds are systematically much better for Matula algorithm than for Welsh-Powell algorithm.
We should just remember that these values are actually upper bounds on the split-chromatic and cochromatic numbers, however the number of used colors are generally much below these values as shown by the first numerical experiments. In [5], computations on random graphs with known split-chromatic number[6] show that these algorithms perform rather well in providing initial solutions to some possible heuristics. Our purpose here is just to study these properties from a theoretical point of view.

6.7 Concluding remarks

We studied the approximation behavior of Min Split-coloring and Min Cocoloring in some classes of graphs where they are shown to be $\mathcal{NP}$-hard. We showed that both problems admit the same non-approximability results as Min Coloring for the standard approximation ratio. Nevertheless, they are better approximated from the differential approximation ratio point of view since they both admit a DPTAS, whereas Min Coloring can be approximated only with a constant differential ratio. As a future work, subclasses of graphs could be characterized where these problems admit better approximations; for instance, the case of edge cocoloring could be handled. In Chapter 7 we will develop a better approximation algorithm for a particular case of Min Split-coloring in permutation graphs.
On the other side, the detection of a class of graphs where Min Cocoloring is $\mathcal{NP}$-hard and Min Split-coloring is polynomially solvable remains an open problem; however, following Corollary 6.16, from now on we can narrow our research in a way excluding the classes of graphs, where Min Cocoloring is shown to be reduced to Min Split-coloring. Note that this latter class contains perfect graphs. In the next chapter, we will give more results on classes

[6] Such graphs are constructed in [45]; first a graph $G = kK_k$ is taken and a k-split-coloring of kK_k with n_c cliques and n_s stable sets such that $\max(n_c, n_s) = k$ is fixed. Then, new vertices are added to G in such a way that each new vertex forms either a clique with an already fixed clique or a stable set with an already fixed stable set; this is done in a random fashion. At the end, we obtain a graph G for which there is a k-split-coloring and moreover $\chi_S(G) = k$, since $kK_k \subseteq G$.

of graphs where both problems may behave differently in terms of complexity classes they belong to (see Section 7.4).
An interesting future research direction arising from this chapter, would be to characterize classes of graphs for which the Welsh-Powell or Matula type algorithms respectively described in Sections 6.6.2 and 6.6.3 give an optimal split-coloring and/or an optimal co-coloring.

Chapter 7

Permutation graphs

In this chapter, based on [42], we handle generalized coloring problems, and more specifically Min Split-coloring, in permutation graphs. Given a permutation $\pi(N)$ of N numbers, the corresponding *permutation graph* is obtained by representing each number by a vertex and linking vertices i and j with an edge whenever we have $i > j$ and $\pi^{-1}(i) < \pi^{-1}(j)$, where $\pi^{-1}(i)$ is the position of number i in π. A permutation graph can also be seen as a graph G such that G and its complement $\overline{G}$ are comparability graphs; hence permutation graphs are a subclass of perfect graphs. Remark also that permutation graphs contain cographs as a subclass since cographs can be alternatively defined as comparability graphs of multitrees [93].

Note that, in permutation graphs, stable sets correspond to increasing sequences, and cliques to decreasing sequences. Partitioning permutations into increasing and decreasing sequences has been extensively studied [117, 96, 18]. Our work will extend this research and raise some related open questions. Observe that a minimum cocoloring is a partition of a permutation into a minimum (total) number of increasing and decreasing subsequences. Also, an optimal split-coloring of a permutation graph corresponds to a partition of a given permutation into a minimum number of combinations of an increasing and a decreasing subsequences. Let us first recall that we have already established, in Corollary 6.17, the $\mathcal{NP}$-hardness of Min Split-coloring in permutation graphs, by using the fact that Min Cocoloring is also $\mathcal{NP}$-hard [117] in permutation graphs.

In what follows, first we give two applications of Min Split-coloring in permutation graphs; one for sorting cars of a train and the other one in robotics. This latter application gives rise to a new problem, called Min Ordered Collecting, in permutations (rather than in permutation graphs) which is slightly different from Min Split-coloring and which will be studied in more details in Section 7.2. We will observe that Min Ordered Collecting remains $\mathcal{NP}$-hard by using a result of [110]. Nevertheless, some related polynomial cases will be established; namely, recognition of 2-lower (or upper) unimodal permutations and finding a maximum l-modal subsequence in a given permutation[1]. The latter is used, in Section 7.2.4, to derive

[1] The link between Min Ordered Collecting and (lower/upper) (uni-/l-) modal partitions of permutations will also be established in the corresponding section.

a differential approximation scheme for the problem of covering a given permutation by a minimum number of l-modal subsequences. In Section 7.3, we introduce the problem, called Min Threshold-coloring, of covering a permutation graph by a minimum number of threshold graphs. This problem is closely related to, but different from, Min Ordered Collecting in permutations. We show that Min Cocoloring reduces to Min Threshold-coloring under exactly the same conditions as for Min Split-coloring (expressed in Theorem 6.15). Then, in Section 7.4, we show that Min Split-coloring, Min Threshold-coloring and Min Cocoloring are $\mathcal{NP}$-hard in triangle-free graphs. Note that, according to Theorem 6.15, triangle-free graphs could have been a class of graphs in which Min Split-coloring is polynomially solvable and Min Cocoloring is $\mathcal{NP}$-hard; we show that this is not the case. This is obtained as a consequence of a more general result comparing computational difficulties of the problems under consideration.

7.1 Applications of permutation graphs

7.1.1 Sorting cars

Consider the problem where there are on an input track cars labeled with numbers in $N = \{1, \ldots, n\}$ and they are linearly ordered so that their labels form a permutation $\pi(N)$. The objective is to use a minimum number of parallel tracks in order to have all the cars sorted in increasing order at the output track. This problem can be modeled as a minimum coloring problem in the permutation graph $G(\pi)$ corresponding to $\pi(N)$. Then the problem consists in partitioning $G(\pi)$ into a minimum number $\chi(G(\pi))$ of stable sets, i.e., increasing sequences in π, so that cars could be reassembled on the exit side of the tracks to obtain one increasing sequence (see [80] for details).
The disadvantage of this method is that one may be obliged to use more tracks than the number of tracks available. In such a situation, one may think of other methods to solve this problem. A solution consists of using a cocoloring or (better) a split-coloring instead of a coloring since we have $\chi_S(G) \leq z(G) \leq \chi(G),\ \forall G$. The only requirement on the railway net is to have an additional loop at the beginning of the output track (see Figure 7.1) allowing the reversal of decreasing sequences corresponding to cliques into a cocoloring or a split-coloring. In such a railway net, each one of the parallel tracks contains an increasing sequence (a stable set) or a decreasing sequence (a clique) for a cocoloring, and an increasing sequence mixed with a decreasing sequence (a split graph) for a split-coloring. Then in the latter case, we rearrange all tracks to obtain only one increasing sequence by track; cars in the decreasing sequence are sent to the loop, while cars in the increasing sequence go directly through the straight track so that, when bringing back the cars to their original track, we can obtain only one increasing sequence. The same procedure can be applied just to reverse decreasing sequences in a cocoloring. Once we obtain only increasing sequences in each track with two moves (one forward and one back), cars can be easily ordered at the exit by pulling off each time the car with the smallest remaining label among the first cars

of each track.

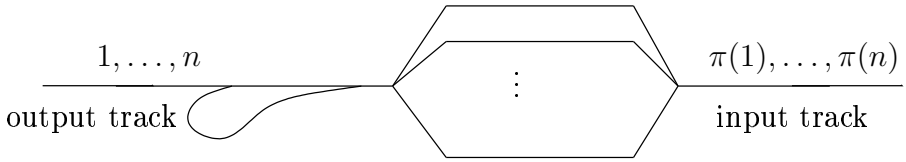


Figure 7.1: Sorting cars.

As mentioned, both Min Split-coloring and Min Cocoloring are $\mathcal{NP}$-hard in permutation graphs. Therefore the use of heuristics or approximation algorithms as the ones given in Chapter 6 is needed to solve the above problem.

7.1.2 Robots collecting items in the order of decreasing sizes

Suppose that some items to be collected are aligned along a storage corridor; each item i has a certain label $\pi(i)$ in $N = \{1, \ldots, n\}$ and hence the labels of the items form a permutation $\pi(N)$. We may consider that labels are inversely proportional to the sizes of the items. There is a robot to collect these items with the constraints that the robot makes a two-way trip along the corridor, and that the sizes (respectively labels) of the items collected should be decreasing (respectively increasing) for the whole trip of the robot to ensure the stability of the pile. The robot can start from either left or right. The objective is to minimize the number of two-way trips the robot makes in order to collect all the items in a way satisfying the constraint of increasing labels.
One can observe that this problem corresponds to a restricted version of Min Split-coloring in permutation graphs. The labels of the items form a permutation of $N = \{1, \ldots, n\}$. Then, for a two-way trip starting from left (respectively right), the items collected in the forward trip correspond to a stable set (respectively clique), and the items collected in the way back to a clique (respectively stable set) in the corresponding permutation graph. In addition, each split graph (equivalently each two-way trip) has a split partition such that the smallest label in the clique is greater than the largest label of the stable set; we will see, in Section 7.3, that this problem is closely related to covering the vertices of a permutation graph by a minimum number of threshold graphs. First, let us introduce the appropriate terminology for the above problem and also establish some complexity results.

7.2 Min Ordered Collecting

7.2.1 Terminology and $\mathcal{NP}$-hardness

Definition 7.1 (feasible trip). *Given a permutation π, we call feasible trip, a subsequence of π which is an increasing sequence for a whole two-way trip of the robot starting from either left or right.*

Definition 7.2 (Min Ordered Collecting). *Min Ordered Collecting is the problem of covering π by a minimum number of feasible trips. The optimal value is denoted by $\rho(\pi)$.*

Also, we denote by $\rho_l(\pi)$ and $\rho_r(\pi)$ the values of optimal ordered collectings where all feasible trips start from left and respectively right.
Before mentioning the complexity status of Min Ordered Collecting, let us define the notion of l-modal sequences, introduced in [110].
We call an *internal extremum* of a permutation π, a number $\pi(i)$ such that $2 \leq i \leq n-1$ and we have either $\pi(i) < \pi(i-1)$ and $\pi(i) < \pi(i+1)$, or $\pi(i) > \pi(i-1)$ and $\pi(i) > \pi(i+1)$. A sequence is *l-modal* if it has at most l internal extrema, the first being of either type. If the first extremum is a maximum then the sequence is called *upper l-modal* and otherwise *lower l-modal*. In the case $l = 1$, we say that a sequence is (upper or lower) *unimodal*. Finally, if a permutation can be partitioned into p decreasing and k increasing subsequences then it is called *p-decreasing k-increasing*.
Let $\pi = (\pi(1), \ldots, \pi(n))$ be a permutation, the *reversal* of π, denoted by $\bar{\pi}$, is the permutation $(\pi(n), \ldots, \pi(1))$. The *inverse permutation*, denoted by π^{-1}, is given by $(\pi^{-1}(1), \ldots, \pi^{-1}(n))$. Finally, we call the *symmetric* of π, the permutation $\pi_s = (n+1-\pi(1), \ldots, n+1-\pi(n))$. Clearly, $\rho_l(\pi) = \rho_r(\bar{\pi})$ and $\rho_r(\pi) = \rho_l(\bar{\pi})$. One can notice that the symmetric and the reversal operations conserve ρ, i.e., $\rho(\pi) = \rho(\bar{\pi}) = \rho(\pi_s)$, since both interchange increasing and decreasing subsequences. Note also that $\rho_l(\pi) = 1$ if and only if π^{-1} is upper unimodal, and $\rho_r(\pi) = 1$ if and only if π^{-1} is lower unimodal; moreover $(\pi^{-1})^{-1} = \pi$. Finally, it can be easily checked that π is upper l-modal if and only if π_s is lower l-modal and vice versa. To illustrate, consider the permutation $\pi = (1, 5, 2, 4, 3)$ which is a feasible trip from left; then $\pi^{-1} = (1, 3, 5, 4, 2)$ is upper unimodal and $(\pi^{-1})_s = (5, 3, 1, 2, 4)$ is lower unimodal.
Using the above terminology, one can state that minimizing feasible trips covering a given permutation π is equivalent to finding the minimum k such that there are at most k unimodal subsequences covering π^{-1}. If we call *Min l-modal*, *Min upper l-modal* and *Min unimodal* the problems of covering a given permutation with a minimum number of respectively l-modal, upper l-modal and unimodal subsequences, then the following theorem of [110] establishes the $\mathcal{NP}$-hardness of all these problems.

Theorem 7.3 ([110]). *Min l-modal, Min upper l-modal and Min unimodal are $\mathcal{NP}$-hard even for fixed l, in particular for $l = 1$.* □

Corollary 7.4. *Given a permutation π, it is $\mathcal{NP}$-hard to find $\rho(\pi), \rho_l(\pi)$ and $\rho_r(\pi)$.*

Proof. The corollary follows from the permutation π^{-1} transforming (in linear time) a permutation π such that $\rho(\pi) = k$ to a permutation which can be covered by k unimodal subsequences; moreover feasible trips from left (respectively right) in π become upper (respectively lower) unimodal sequences in π^{-1}. □

In what follows, we formulate our results in terms of Min (upper/lower) l-modal problem rather than feasible trips. A sequence that can be covered by k (upper/lower) l-modal subsequences will be called *k-(upper/lower) l-modal* and similar notations hold for the unimodal

case. Following subsections contain polynomial time algorithms for some problems related to l-modal sequences.

7.2.2 Recognition of 2-lower unimodal permutations

Let π be a 2-lower unimodal permutation of n consecutive numbers with two lower unimodal subsequences $L_1 = (m_1, \ldots, m_p, \ldots, m_{l_1})$ and $L_2 = (m'_1, \ldots, m'_h, \ldots, m'_{l_2})$ (m_i's and m'_j's are values) covering together π and where p and h are such that $m_p = \min_{1 \leq i \leq l_1}(m_i)$ and $m'_h = \min_{1 \leq j \leq l_2}(m'_j)$. Without loss of generality, we assume that $m_p < m'_h$ and $\pi^{-1}(m_p) < \pi^{-1}(m'_h)$ since in the opposite case, i.e., in the case where the minimum between m_p and m'_h comes after the other one in π, we may work on $\bar{\pi}$ that remains 2-lower unimodal and verifies the above hypothesis. We have two cases illustrated in Figure 7.2 where lower unimodal subsequences are represented as continuous lines on which discrete points are located. In fact, without loss of generality, we may consider only Case 1 since any 2-lower unimodal cover (L_1, L_2) of Case 2 can be seen as Case 1 with two lower unimodal subsequences (L'_1, L'_2) obtained by exchanging some appropriate points between L_1 and L_2: L'_1 will contain L_1 from the first position to the position $\pi^{-1}(m'_h) - 1$ and also L_2 from the position $\pi^{-1}(m'_h)$ to the end; L'_2 is then the remaining subsequence.

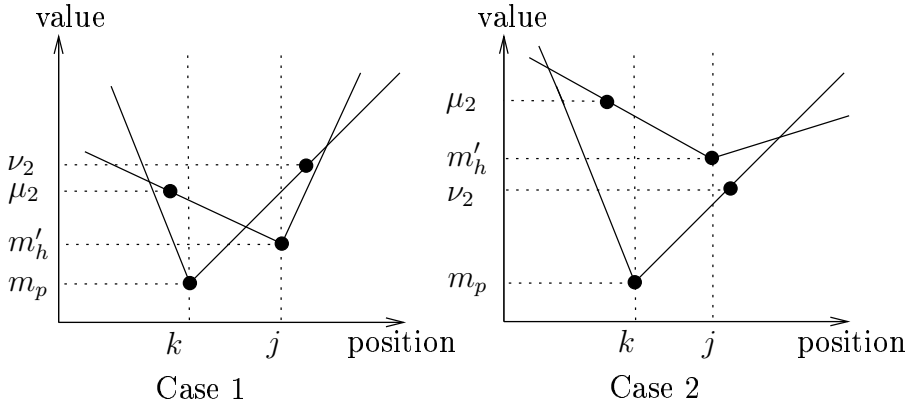


Figure 7.2: 2-lower unimodal permutation.

To recognize a 2-lower unimodal permutation, the idea is to determine two numbers m_p and m'_h that can be considered as minima of two lower unimodal subsequences. Then, for any 2-lower unimodal permutation, $(\pi(1), \ldots, \pi(\pi^{-1}(m_p)))$ is 2-decreasing, $(\pi(\pi^{-1}(m_p)), \ldots, \pi(\pi^{-1}(m'_h)))$ is 1-decreasing 1-increasing and $(\pi(\pi^{-1}(m'_h)), \ldots, \pi\ (n))$ is 2-increasing in such a way that these monotone subsequences can be combined to form two lower unimodal subsequences covering π.

In what follows, $\min(S)$ denotes the minimum value in a subsequence S, and $S_1 \cup S_2$ denotes the concatenation of two subsequences S_1 and S_2. Note that Case 1 of Figure 7.2 is just a possible representation of the two lower unimodal subsequences L_1 and L_2. They can be alternatively drawn in such a way that they intersect only once (between positions k and j). In what follows, we will rather find such a decomposition, if there is one.

Algorithm 21 Recognition of 2-lower-unimodal permutations

Input: a permutation π

Output: 2-lower unimodal cover of π with L_1, L_2 or a negative answer

1. $A \leftarrow$ true;
2. compute $k = \text{argmin}_{i \in \{1,\ldots,n\}}(\pi(i))$;
3. apply the greedy algorithm from left to right to partition $(\pi(1), \ldots, \pi(k))$ into decreasing sequences;
4. **if** two decreasing subsequences (D_1, D_2) partition $(\pi(1), \ldots, \pi(k))$ **then**
5. $\quad \mu_2 \leftarrow \min(D_2)$ and $\pi(k) \leftarrow m_p \in D_1$;
6. **else**
7. $\quad$ **if** A is true **then**
8. $\quad\quad A \leftarrow$ false;
9. $\quad\quad \pi \leftarrow \bar{\pi}$;
10. $\quad\quad$ go to line 2;
11. $\quad$ **else** *//A is false, i.e., we are applying the algorithm to* $\bar{\pi}$
12. $\quad\quad$ **print** "π is not 2-lower unimodal", **exit**.
13. **repeat**
14. $\quad$ apply one step of the greedy algorithm from right to left to find 2 increasing subsequences (from left to right) (I_1, I_2);
15. **until** (Case = 1 : we reach the position k) or (Case = 2 : we reach a position r such that $\pi(r)$ can be put neither in I_1 nor in I_2)
16. **if** Case = 1 **then**
17. $\quad L_1 \leftarrow D_1 \cup I_1$;
18. $\quad L_2 \leftarrow D_2 \cup I_2$;
19. $\quad$ **return** (L_1, L_2).
20. **else** *//Case = 2*
21. $\quad m'_h \leftarrow \min(I_1)$;
22. $\quad \pi^{-1}(m'_h) \leftarrow j$;
23. $\quad I_2 \leftarrow I_2$ from position $j+1$ to n;
24. $\quad \nu_2 \leftarrow \min(I_2)$;
25. $\quad$ **if** $(\mu_2, \pi(k), \ldots, \pi(j), \nu_2)$ can be decomposed into a decreasing and an increasing subsequence (D, I) **then**
26. $\quad\quad$ **if** $\pi(j) \in I$ **then**
27. $\quad\quad\quad$ interchange I_1 and I_2;
28. $\quad\quad L_1 \leftarrow D_1 \cup I \cup I_2$;
29. $\quad\quad L_2 \leftarrow D_2 \cup D \cup I_1$;
30. $\quad\quad$ **return** (L_1, L_2).
31. $\quad$ **else**
32. $\quad\quad$ go to line 7;

Theorem 7.5. *Algorithm 21 decides in time $\mathcal{O}(m + n \log n)$ whether a given permutation π is 2-lower unimodal or not.*

Proof. First of all, note that greedy algorithms finding (I_1, I_2) and (D_1, D_2) assign each number of π to the set with smallest possible index. It follows that I_1 will always be under I_2 and D_1 under D_2 in a lattice representation as in Figure 7.2. That is why D_1 contains necessarily m_p and the greedy algorithm for finding (I_1, I_2) succeeds in finding a number m'_h that can be considered, without loss of generality, as the minimum we are looking for. Furthermore, let us recall the following fact (see for instance [80]), which shows that these greedy algorithms never fail in finding (I_1, I_2) and (D_1, D_2). For the sake of completeness, we give a short proof.

Claim 7.6. *For a given permutation π, the greedy coloring algorithm based on the order $\pi(1), \pi(2), \ldots$ of the vertices gives an optimal coloring of $G(\pi)$.*

Proof. (of the Claim) This can also be viewed as a consequence of the fact that permutation graphs belong to the class of *perfectly orderable graphs* defined by the existence of a perfect order of their vertices. The greedy coloring algorithm based on this order gives an optimal coloring. It is known that an order is *perfect* if it induces no P_4 with edges ab, bc and cd, where $a < b$ and $d < c$ in the order [25]. Here, as it can be observed in Figure 7.3, for any induced P_4 on ab, bc and cd, we have in the order $\pi(1), \pi(2), \ldots$ either $b < a$ or $c < d$.

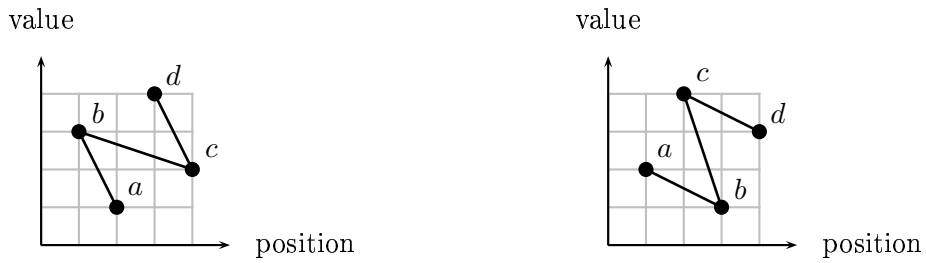


Figure 7.3: Induced P_4's in permutations.

□

If π is 2-lower unimodal, then after finding (D_1, D_2) and (I_1, I_2) as described in the algorithm, one of the two following things may happen; either (D_1, D_2) and (I_1, I_2) cover together π, then $L_1 = D_1 \cup I_1, L_2 = D_2 \cup I_2$ gives a 2-lower unimodal cover for π (it corresponds to the case where $m'_h = \min(\pi(k-1), \pi(k+1))$ which is expressed by Case = 1 in line 15); or there are numbers between positions k and j which are not covered by (D_1, D_2) and (I_1, I_2) (it corresponds to Case = 2 in line 15). In the latter case, the subsequence $(\mu_2, \pi(k), \ldots, \pi(j), \nu_2)$ should be 1-decreasing 1-increasing (D, I) in such a way that combining them with (D_1, D_2) and (I_1, I_2) gives two lower unimodal subsequences. Note that for every such decomposition (D, I), $m_p \in I$ (since $m_p = \min_{1 \leq i \leq n}(\pi(i))$), and $\mu_2 \in D$ since

$\pi^{-1}(\mu_2) < \pi^{-1}(m_p)$ and $\mu_2 > m_p$. Furthermore, if $\pi(j) \in D$ then necessarily $\nu_2 \in I$ (since $m'_h < \nu_2$ and $\pi^{-1}(m'_h) < \pi^{-1}(\nu_2)$) and $L_1 = D_1 \cup I \cup I_2, L_2 = D_2 \cup D \cup I_1$ are two lower unimodal subsequences covering π (see lines 28 and 29). Otherwise, if $\pi(j) \in I$ then I can be combined with I_1 and D with I_2, hence we obtain two lower unimodal subsequences L_1 and L_2 as described by interchanging I_1 and I_2 as it is done in line 27. If the algorithm fails to find such decompositions ((D_1, D_2) and (D, I)) then we should apply the same on $\bar{\pi}$ which is necessarily as in Case 1 of Figure 7.2. If Algorithm 21 gives no 2-lower unimodal decomposition for $\bar{\pi}$ neither then it means that π is not 2-lower unimodal.
As for the complexity of Algorithm 21, the greedy algorithms take time in $\mathcal{O}(m)$ and finding one decreasing one increasing decomposition takes time in $\mathcal{O}(n \log n)$ [18]. □

Corollary 7.7. *Given a permutation π, it can be decided in time $\mathcal{O}(m + n \log n)$ whether it is 2-upper unimodal or not.*

7.2.3 Maximum l-modal subsequence

Consider a permutation π which is upper or lower l-modal. Then, it can be easily checked that π^{-1} corresponds to an increasing subsequence of labels representing items collected during $l+1$ trips back and forth of a robot (starting from left and right, respectively) which does not unload its charge until the end of $l + 1$ trips and which obeys the constraint of increasing labels during its whole trip. In other words, a robot can collect all the items ordered according to π^{-1} with $l+1$ back and forth trips respecting the constraint of increasing labels. In what follows, given a permutation π representing the sizes of the items, we show how to collect a maximum number of items by $l + 1$ trips of a robot. This will be explained in terms of maximum l-modal subsequence of a given permutation.

Theorem 7.8. *Given a permutation π, a maximum upper or lower l-modal subsequence of π can be found in time $\mathcal{O}(n \log n)$ for fixed l, and in time $\mathcal{O}(n^2 \log n)$ for arbitrary l.*

Proof. Let $\pi = (\pi(1), \ldots, \pi(n))$ be a permutation in which we want to find a maximum upper l-modal subsequence. We construct a permutation π^l consisting of $\lfloor l/2 \rfloor + 1$ copies of π and $\lceil l/2 \rceil$ copies of $\pi_s = (n+1-\pi(1), \ldots, n+1-\pi(n))$ piled up alternatively and in such a way that they are shifted one step to the left at each layer. Let us call π_j the permutation at layer j, then the values in π_j are not consecutive; there are $l-1$ empty positions between consecutive values in π_j. The permutation π^l is given by all π_j for $j = 0, \ldots, l$ and contains values from 1 to $(l + 1)n$. Then the following formula defines each π^l:

$$\text{for } j = 0, \ldots, l, \qquad \pi_j(i + il - j) = \begin{cases} \pi(i) + jn & \text{if } j \text{ is even} \\ n + 1 - \pi(i) + jn & \text{if } j \text{ is odd} \end{cases}, \quad i = 1, \ldots, n$$

An example of construction of π^l can be found in Figure 7.4, where $\pi = (4, 2, 3, 6, 5, 7, 1, 8)$ and $l = 2$. Here, a maximum upper 2-modal subsequence is $(2, 3, 6, 5, 1, 8)$. It corresponds to the increasing subsequence $(2, 3, 6, 12, 17, 24)$ in π^2.

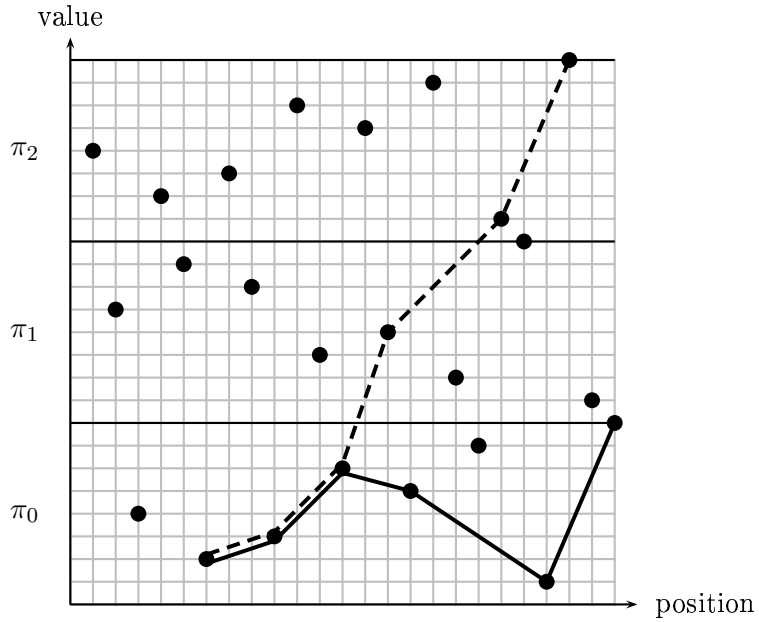


Figure 7.4: Maximum upper 2-modal subsequence

Now, one can observe that an increasing subsequence in π^l corresponds to an upper l-modal subsequence of the same size in π and vice versa. In fact, as can be seen in the example of Figure 7.4, an increasing subsequence in π_j corresponds to a decreasing subsequence in π if j is odd, and to an increasing subsequence if j is even. Moreover, due to the shift of one step to the left at each layer, copies of one label in different π_j's cannot be taken together in one increasing subsequence of π^l; hence, each number in π is represented at most once in any increasing subsequence of π^l. Then, the result follows from the fact that a maximum increasing subsequence in π^l can be found in time $\mathcal{O}((l+1)n\log((l+1)n))$. As for a maximum lower l-modal subsequence, it is sufficient to consider the symmetric of π since it interchanges upper and lower l-modal subsequences. □

7.2.4 Differential approximation

In [110], the question of existence of approximation schemes for Min Cocoloring (in permutation graphs) and Min l-modal was left open. On the other hand, some approximation algorithms with constant approximation ratios are given for the same problems. We have already stated in Theorem 6.21 that, given an integer p, there is a polynomial algorithm in time $\mathcal{O}(n^{3p+1})$ guaranteeing a differential approximation ratio of $(1-1/p)$ for Min Cocoloring in arbitrary graphs. Here, we answer the open question on the approximability of Min l-modal from a different point of view; we give an approximation scheme using differential approximation ratio instead of the classical ratio. It is obtained by a modification of Algorithm 19 given in Section 6.5.2.

Algorithm 22 DPTAS-l-modal

Input: a permutation π of size n, an integer p

Output: an l-modal covering of π with differential approximation ratio of $(1 - 1/p)$

1. **while** the current permutation has a maximum l-modal subsequence of size at least $p(l + 2)$ **do**
2. color such an l-modal subsequence with a new color;
3. complete the solution by an exhaustive search on the remaining permutation.

Theorem 7.9. *Algorithm 22 is a $\mathcal{O}(n^2 \log n)$ time algorithm guaranteeing a differential approximation ratio of $(1 - 1/p)$ for Min l-modal.*

Proof. The proof is similar to the one of Theorem 6.21 with the only difference that for a given permutation π of size n, we have $\omega(\pi) = \lceil n/(l+2) \rceil$ since any subsequence of size $l+2$ is necessarily l-modal. The complexity is implied by the complexity of finding a maximum l-modal subsequence given in Theorem 7.8, which is repeated at most $\lceil n/(l+2) \rceil$ times, and by the fact that the second step takes only constant time; this holds since the number of vertices in the remaining graph is bounded by a constant given by a Ramsey number (since there is no clique or stable set of size $p(l + 2)$ in the remaining graph). □

Furthermore, in exactly the same way as for Min Cocoloring and Min Split-coloring in general graphs, since Min l-modal has integral values and $\omega(\pi) - \beta(\pi)$ is polynomially bounded, one can immediately deduce that there is no DFPTAS for Min l-modal.

7.3 Min Threshold-coloring

We have already mentioned that feasible trips in a given permutation π, correspond to particular split graphs in the graph $G(\pi)$. In this section, we explore the links between feasible trips (or unimodal subsequences) and split graphs. We show that feasible trips correspond more precisely to threshold graphs which form a subclass of split graphs. Nevertheless, a minimum ordered collecting ρ is a uniquely defined parameter in permutations and not in permutation graphs. These results give birth to the idea of considering the problem of covering the vertices of a given graph with a minimum number of threshold graphs; this problem is then studied together with Min Split-coloring and Min Cocoloring in terms of complexity class it belongs to in restricted classes of graphs.
The following notion is given in [80].

Definition 7.10 (shuffle product). *Let σ and τ be two sequences. The shuffle product of σ and τ is defined as follows:*

$$\sigma \sqcup\!\sqcup \tau = \{\sigma_1\tau_1 \dots \sigma_k\tau_k : \sigma = \sigma_1 \dots \sigma_k \text{ and } \tau = \tau_1 \dots \tau_k\}$$

where σ_i and τ_i are subsequences, k ranges over all integers and the juxtaposition means concatenation.

Now, let us observe the following fact.

Proposition 7.11. *Let π be a permutation, then:*

$$\rho_l(\pi) = 1 \quad \Leftrightarrow \quad \pi \text{ is a shuffle product of type 1} : [1, \ldots, p] \sqcup\!\sqcup [n, \ldots, p+1], 1 \leq p < n;$$
$$\rho_r(\pi) = 1 \quad \Leftrightarrow \quad \pi \text{ is a shuffle product of type 2} : [p+1, \ldots, n] \sqcup\!\sqcup [p, \ldots, 1], 1 \leq p < n;$$

where shuffle products can also have only one term.

Proof. Clearly, a shuffle product of type 1 (respectively 2) gives the program of a feasible trip for the robot which starts from left (respectively right); the first (respectively second) term in the shuffle product coincides with the items to be collected in the way forward, and the second (respectively first) term with the items to be collected in the way back. Conversely, given a feasible trip from left (respectively right), the increasing subsequence in the feasible trip gives the first term of a shuffle product of type 1 (respectively type 2) and the decreasing subsequence the second term. □

Recall that a graph G is a *threshold graph* if it is a split graph with partition (K, S) and the neighborhoods of vertices in S are nested, i.e., one can label vertices in S with integers such that if $i < j$ then $N(i) \subseteq N(j)$. The relationship between threshold graphs and feasible trips is derived from the following theorem of [79].

Theorem 7.12 ([79]). *The threshold graphs are precisely those permutation graphs corresponding to sequences contained in*

$$[1, 2, \ldots, p] \sqcup\!\sqcup [n, n-1, \ldots, p+1]$$

where p and n are positive integers. □

An important consequence of this theorem is the following: if π is a shuffle product (hence a feasible trip) then $G(\pi)$ is a threshold graph but if G is a threshold graph then there exists a permutation π which is a shuffle product and such that G is isomorphic to $G(\pi)$. In other words, there may exist several permutations giving a permutation graph isomorphic to G. It follows that there may be different permutations π_1 and π_2 such that $G(\pi_1)$ is isomorphic to $G(\pi_2)$, and π_1 is a feasible trip but not π_2. Take as example $\pi_1 = (1, 4, 2, 3)$ and $\pi_2 = (3, 1, 2, 4)$. Then $G(\pi_1)$ and $G(\pi_2)$ are isomorphic to a graph G consisting in a path on tree vertices and one isolated vertex. G is obviously a threshold graph and $\rho(\pi_1)$=1 but $\rho_l(\pi_2) \neq 1$ and $\rho_r(\pi_2) \neq 1$. This observation shows that, given a permutation π, Min Ordered Collecting of π is not equivalent to the problem of covering $G(\pi)$ by a minimum number of threshold graphs, since $\rho(\pi)$ is an invariant of the permutation and not an invariant of the corresponding permutation graph.
Following the above remark, a natural extension of Min Ordered Collecting in graphs appears as the following problem.

Definition 7.13 (Min Threshold-coloring). *Min Threshold-coloring is the problem of covering the vertices of a given graph G by a minimum number of threshold graphs. The optimal value, called threshold-chromatic number, is denoted by $\chi_T(G)$.*

In what follows, we discuss its complexity compared to Min Split-coloring and Min Cocoloring.

Theorem 7.14. *Let $\mathcal{G}$ be a class of graphs closed under addition of disjoint cliques without link to the rest of the graph and under addition of a complete k-partite graph completely linked to the rest of the graph. Then, Min Cocoloring reduces in polynomial time to Min Threshold-coloring in the class $\mathcal{G}$.*

Proof. Consider a graph $G = (V, E)$ in $\mathcal{G}$ of size n as an instance of Min Cocoloring. We obtain an instance $G'_{i,j}$ of Min Threshold-coloring from G by adding iK_n, i.e., i disjoint cliques (called *external cliques*) each of size n without any link to the rest of the graph, and j stable sets (called *external stable sets*) each of size n with complete links to G, to iK_n and between themselves, hence $G'_{i,j} \in \mathcal{G}$.
Suppose there is an optimal cocoloring of G with p cliques and k stable sets; so that $z(G) = p + k$. Let us analyze what happens while considering Min Threshold-coloring in $G'_{k,p}$. Suppose there is an optimal threshold-coloring with the following threshold graphs (cliques and stable sets with all vertices in G are called *internal*):

- l external cliques matched with l external stable sets,
- h internal cliques matched with h internal stable sets,
- l' isolated external stable sets (as threshold graphs),
- l'' isolated external cliques (as threshold graphs),
- $p - l - l'$ internal cliques matched with $p - l - l'$ external stable sets,
- $k - l - l''$ external cliques matched with $k - l - l''$ internal stable sets.

Therefore, we have $\chi_T(G'_{k,p}) = p + k + h - l$. In addition, one can write $\chi_T(G'_{k,p}) \leq p + k$ since k external cliques matched with k internal stable sets, and p external stable sets matched with p internal cliques form a threshold-coloring of value $p + k$. Thus, we have $h \leq l$. On the other hand, this threshold-coloring provides a cocoloring of G of value $2h + p - l - l' + k - l - l'' \geq p + k$ because $p + k$ is the value of an optimal cocoloring. Since $h \leq l$, it follows that $l' = l'' = 0$; there is no isolated clique or stable set taken as a whole threshold graph in any optimal threshold-coloring. Consequently, we have $h = l$ which implies that $\chi_T(G'_{k,p}) = p + k = z(G)$. Hence, one can solve Min Threshold-coloring in $G'_{i,j}$ for different values of i and j (at most $\chi(G) \times \theta(G)$ values to check) and pick the one for which $\chi_T(G'_{i,j})$ is equal to the value of cocoloring of G induced by that threshold-coloring; this will be an optimal cocoloring of G. Equivalently, one can choose a pair (p, k) giving

an optimal cocoloring by detecting the maximum value of $\chi_T(G'_{i,j})$ with respect to (i,j); clearly $\chi_T(G'_{i,j})$ is always less than or equal to the value of the cocoloring of G induced by this solution and the equality will be obtained for (i,j) giving an optimal cocoloring. This reduction is given in Algorithm 23. □

Algorithm 23 Polynomial reduction of Min Cocoloring to Min Threshold-coloring

Input: a graph G as an instance of Min Cocoloring, an exact algorithm for Min Threshold-coloring

Output: an optimal cocoloring of G

1. **for all** $(i,j) \in \{1,\ldots,n\} \times \{1,\ldots,n\}$ **do**
2. construct $G'_{i,j} = (G \cup iK_n) \oplus \overline{jK_n}$;
3. compute an optimal threshold-coloring $\mathcal{TC}$ of $G'_{i,j}$;
4. **if** the value of the cocoloring $\mathcal{Z}$ in G induced by $\mathcal{TC}$ is equal to $\chi_T(G'_{i,j})$ **then**
5. **return** cocoloring $\mathcal{Z}$ of G.

Corollary 7.15. *Min Threshold-coloring is $\mathcal{NP}$-hard in permutation graphs.*

Let us mention a result in [89] stating that coloring a graph in such a way that color classes are P_4-free is $\mathcal{NP}$-complete in comparability graphs. Knowing that threshold graphs are exactly the $(P_4, 2K_2, C_4)$-free graphs, Corollary 7.15 can also be interpreted in the following way: coloring a graph, where color classes are $(P_4, 2K_2, C_4)$-free, is $\mathcal{NP}$-complete in permutation graphs.

7.4 More complexity results

According to Theorem 6.15, Min Split-coloring is at least as difficult as Min Cocoloring in most (but not all) of the "natural" classes of graphs. Note that the contrary is not excluded; in Proposition 5.9, we describe a class of graphs, denoted by $\mathcal{G} = \{G \cup nK_{2n}\}$, where Min Split-coloring is polynomially solvable while Min Cocoloring is $\mathcal{NP}$-hard. This class of graphs is obtained from any arbitrary graph G of size n by adding n disjoint K_{2n} not linked to the rest of the graph. Clearly, it is neither a "natural" nor a hereditary class. Hence, the existence of "natural" classes of graphs where Min Split-coloring is polynomially solvable while Min Cocoloring is $\mathcal{NP}$-hard remains an interesting research topic. The same question can be also asked for Min Threshold-coloring since Theorem 7.14 is the threshold counterpart of Theorem 6.15. One could think of the class of *triangle-free graphs*, i.e., graphs having no induced triangle, to which neither Theorem 6.15 nor Theorem 7.14 apply, as a candidate for such a class of graphs. We show, by means of results on a larger class of graphs containing triangle-free graphs, that this is not the case. Let us first state a result of Gimbel et al. in [78].

Lemma 7.16 ([78]). *If $\mathcal{G}$ is a class of graphs closed under taking disjoint unions then the $\mathcal{NP}$-completeness of k-coloring on $\mathcal{G}$ implies the $\mathcal{NP}$-completeness of k-cocoloring on $\mathcal{G}$.* □

This result, combined with the $\mathcal{NP}$-completeness of k-coloring in triangle-free graphs (see [107]), implies immediately that k-cocoloring is $\mathcal{NP}$-complete in triangle-free graphs. Here, we show moreover that the cochromatic number of a triangle-free graph is equal to its chromatic number.

Proposition 7.17. *For any triangle-free graph G, we have $z(G) = \chi(G)$.*

Proof. First, let us introduce perfect cochromatic graphs as defined in [123]; a graph G is *perfect cochromatic* if $z(G') = \min(\chi(G'), \chi(\overline{G'}))$ for all subgraphs G' of G. It is easy to verify that triangle-free graphs are perfect cochromatic since all subgraphs in the list of forbidden subgraphs characterizing perfect cochromatic graphs contain an induced triangle [123]. We will show that $\chi(G) \leq \chi(\overline{G})$ which gives $z(G) = \min(\chi(G), \chi(\overline{G})) = \chi(G)$ for all triangle-free graphs of size greater than or equal to 3. Observe that for a triangle-free graph G of size at least 3, we have $\chi(\overline{G}) \geq \lceil n/2 \rceil$ since every stable set of $\overline{G}$ has at most two vertices. We only have to show that $\chi(G) \leq \lceil n/2 \rceil = \lfloor (n+1)/2 \rfloor$ for a triangle-free graph G of size $n \geq 3$. Without loss of generality, we may assume that G is k-critical. There exists a k-coloring $(S_1, \ldots, S_k)$ of G such that every vertex $v \in S_i$ is adjacent to at least one vertex in each S_j with $j < i$. Furthermore $|S_k| = 1$ by criticality. In particular, vertex x in S_k is adjacent to some vertex y in S_{k-1} which is itself adjacent to a vertex z in S_{k-2}; but x is also adjacent to a vertex w in S_{k-2}. We must have $w \neq z$, for otherwise G would contain a triangle. This gives $|S_{k-2}| \geq 2$. The same reasoning shows that $|S_i| \geq 2$ for all $i \leq k-2$.

Claim 7.18. *If $|S_i| = 2$ for $i = 1, \ldots, k-2$ and $|S_{k-1}| = 1$, then G is bipartite.*

Proof. (of the Claim) Let $S_i = \{a_i, b_i\}$ for $i = 1, \ldots, k-2$. We can assume that vertex x is adjacent to $a_1, \ldots, a_{k-2}$ and to $y \in S_{k-1}$; then $\{a_1, \ldots, a_{k-2}, y\}$ is a stable set, otherwise G contains a triangle. But then y is adjacent to $b_1, \ldots, b_{k-2}$; hence $\{b_1, \ldots, b_{k-2}, x\}$ is a stable set. Thus G is bipartite[2]. This ends the proof of the claim. □

Since $k \geq 3$, we must have either $|S_i| \geq 3$ for some $i \leq k-2$ or $|S_{k-1}| \geq 2$. In both cases, G has size $n = \sum_{i=1}^{k} |S_i| \geq 2(k-2) + 1 + 1 + 1 = 2k - 1$. Hence $k = \chi(G) \leq \lfloor (n+1)/2 \rfloor$. □

Here are the split and threshold counterparts of Lemma 7.16.

Proposition 7.19. *If $\mathcal{G}$ is a class of graphs closed under taking disjoint unions then the $\mathcal{NP}$-completeness of k-coloring on $\mathcal{G}$ implies both the $\mathcal{NP}$-completeness of k-split-coloring and the $\mathcal{NP}$-completeness of k-threshold-coloring on $\mathcal{G}$.*

Proof. Let $G \in \mathcal{G}$ be an instance of k-coloring. We construct an instance G' of k-split-coloring by taking n disjoint copies of G; $G' = nG$ is obviously in $\mathcal{G}$. It is easy to see that $\chi_S(G') \leq \chi(G') = \chi(G)$. Assume we have t cliques $C_1, \ldots, C_t$ in an optimal split-coloring of nG, where $t < n$. Then $nG \setminus \cup_{i=1}^{t} C_i$ contains $(n-t)G$ as a subgraph, hence $\chi_S(nG) \geq \chi(G)$.

[2] Applying the same argument recursively, one can verify that G is in fact a complete bipartite graph where the edges of the matching $\{a_i b_i, i = 1, \ldots, k-2\}$ are removed.

Therefore, we have $\chi_S(nG) = \chi(G)$ and if we could solve in polynomial time k-split-coloring on nG, then we could also solve in polynomial time k-coloring on G, but this is impossible since it is $\mathcal{NP}$-complete.
The same proof also holds for k-threshold coloring. □

It follows immediately that Min Split-coloring and Min Threshold-coloring are $\mathcal{NP}$-hard in triangle-free graphs.
These results show not surprisingly that, in the class of graphs closed under taking disjoint unions, whenever k-coloring is $\mathcal{NP}$-complete, so are k-cocoloring, k-split-coloring and k-threshold-coloring. As a consequence, classes of graphs where k-cocoloring, k-split-coloring and k-threshold-coloring admit potentially different complexity results can only occur either in classes of graphs closed under taking disjoint unions but where k-coloring can be solved in polynomial time (as it is the case for line-graphs of bipartite graphs (see Theorems 5.4 and 5.8)), or in classes of graphs not closed under taking disjoint unions; this is, for instance, non-hereditary classes of graphs (see Proposition 5.9 on the class of graphs $\mathcal{G} = \{G \cup nK_{2n}\}$) or classes of graphs defined by some disconnected forbidden subgraphs. To complete this picture, let us also recall that, according to Corollary 6.16, in subclasses of perfect graphs, Min Cocoloring cannot be $\mathcal{NP}$-hard while Min Split-coloring is polynomially solvable.
Finally, note that Lemma 7.16 and Proposition 7.19 can also be stated replacing k-coloring by k-clique cover and disjoint union by join; similar arguments are then used in their respective proofs.

7.5 Open questions

There are several research topics arising from this chapter. Although it is in general $\mathcal{NP}$-complete to recognize polar graphs [21], it is shown in Chapter 4 that a polar cograph can be recognized in time $\mathcal{O}(n \log n)$. The question of knowing whether the recognition of polar permutation graphs (which is a class strictly containing cographs) is polynomial or not, is an interesting open problem. A simpler question remains open as well: recognizing $(1,t)$- or $(s,1)$-polar permutation graphs. In return, in the case where s and t are fixed, the polynomial time recognition of (s,t)-polar permutation graphs follows from the following theorem of Brandstädt et al. (Note that this is already implied by Corollary 1.37 with a higher time complexity.)

Theorem 7.20 ([18]). *Given a permutation graph G, for a fixed m, it can be recognized in time $\mathcal{O}(n^m)$ whether G is (p,k)-colorable, where $p+k=m$, or not.* □

Corollary 7.21. *Given a permutation graph G, for fixed s and t, it can be recognized in time $\mathcal{O}(n^{s+t})$ whether G is (s,t)-polar or not.*

Sketch of proof. For fixed p and k, the algorithm in [18] recognizes whether a given permutation graph admits a (p,k)-coloring or not. Roughly speaking, this algorithm enumerates

all possibilities in a clever way, giving a time complexity of $\mathcal{O}(n^{p+k})$. It can be slightly modified to recognize (s,t)-polar permutation graphs by incorporating additional rules, namely complete links between stable sets and absence of links between cliques. It will allow us to consider fewer solutions than we need for the (p,k)-coloring and hence will keep the same time complexity. □

Numerous open questions arise also from the problem of Min Ordered Collecting. We have already pointed out that, for a permutation π, we have $\rho(\pi) \geq \chi_T(G(\pi))$. Given a permutation graph G, is there a permutation π such that $G(\pi)$ is isomorphic to G and $\rho(\pi) = \chi_T(G)$? It would be also interesting to know, for a permutation graph G, how large can be the quantity $\max_{\pi_i} |\chi_T(G) - \rho(\pi_i)|$, where $G(\pi_i)$ is isomorphic to G.
As a natural continuation of our research, one could investigate the relative difficulty of Min Threshold-coloring with respect to Min Split-coloring and Min Cocoloring. A first step in this direction would be to analyze the complexity of Min Threshold-coloring in line graphs of bipartite graphs. Recall that, in this class of graphs, Min Cocoloring is polynomially solvable while Min Split-coloring is $\mathcal{NP}$-hard.
Another interesting topic concerns minimal obstructions.

Definition 7.22 (ideal permutation). *A permutation π is called ideal if we have $\rho(\pi) = \min(\rho_l(\pi), \rho_r(\pi))$.*

Definition 7.23 (minimal obstruction). *A permutation π is a minimal obstruction if it verifies $\rho(\pi) < \min(\rho_l(\pi), \rho_r(\pi))$ and all subpermutations $\pi' \subset \pi$ are ideal, i.e., $\rho(\pi') = \min(\rho_l(\pi'), \rho_r(\pi'))$.*

In [75], lower and upper bounds on the size of minimal obstructions for permutations having $\rho(\pi) = 2$ is studied. It can be easily shown by case enumeration that there is no minimal obstruction of size less than or equal to 7 for permutations such that $\rho(\pi) = 2$; smallest minimal obstructions that are detected have size 8. For instance, $\pi = (1,8,4,3,6,2,7,5)$ is a minimal obstruction; $\rho(\pi) = 2$ where the robot collects the items with labels $(1,5,8)$ by starting from left, and the items with labels $(4,3,6,2,7)$ by starting from right. However, it can be easily checked that $\rho_l(\pi) = \rho_r(\pi) = 3$ and that for all subpermutation $\pi' \subset \pi$, we have $\rho(\pi') = \min(\rho_l(\pi'), \rho_r(\pi')) = 2$. On the other hand, it is also shown by computer enumeration that obstructions of size 9 contain always an obstruction of size 8, that is, there is no minimal obstruction of size 9. These results allow us to construct the following conjecture.

Conjecture 7.24. *All minimal obstructions π with $\rho(\pi) = 2$ are of size 8.*

Naturally, one can try to generalize the above conjecture for minimal obstructions π with $\rho(\pi) = k > 2$. Upper and lower bounds on the sizes of such permutations can be studied and the following question can be analyzed: is there a unique size of minimal obstructions for each $k > 2$ such that $\rho = k$? If yes, can we find it? Knowing whether there are permutations π such that $\rho(\pi) - \min(\rho_l(\pi), \rho_r(\pi)) \geq 2$ would also provide us a hint for answering the above questions.

Conclusion

In this thesis, we have started exploring some untouched problems. Although Min Coloring is a very well known and extensively studied problem, many directions ensuing from this field still remain unexplored. Generalized coloring problems were one of them and our investigation on this topic has turned out to be very fruitful. We could emphasize several results concerning the computational complexity of the generalized coloring problems. We considered these problems in some restricted classes of graphs; with such an analysis, either we designed some polynomial time algorithms or we obtained the proof that the problem under consideration remains intractable even in this simpler case. As expected, Min Split-coloring and Min Cocoloring are significantly more difficult to solve compared to Min Coloring, since they both remain $\mathcal{NP}$-hard in some classes of graphs where Min Coloring is polynomially solvable; for instance, in permutation graphs and in line graphs of bipartite graphs (for Min Split-coloring). Our objective was also to determine the relative difficulties of Min Split-coloring, Min Threshold-coloring and Min Cocoloring; we obtained several results in this direction. In particular, we show that, in a large class of graphs including all perfect graphs, Min Split-coloring and Min Threshold-coloring are more difficult than Min Cocoloring. The multiplication of this kind of results would contribute to a better understanding of the generalized coloring problems. Now, let us summarize the main results of each chapter by emphasizing the open questions arising from them.

In Chapter 1, we introduced generalized coloring problems and we studied some k-split-critical structures as well as some solvable cases for (p, k)-colorability. We motivated the following chapters by pointing out interesting questions to handle in this field. A future research already mentioned in this chapter would be the characterization of 2-split-colorable graphs by an infinite list of forbidden configurations.

Chapter 2 dealt with cacti and triangulated graphs where Min Split-coloring is solved in polynomial time. It would be interesting to study a possible extension of these results to some classes of graphs generalizing triangulated graphs, such as weakly triangulated graphs and quasi-triangulated graphs.

In Chapter 3, we showed by means of a new characterization of cographs that there are nice combinatorial algorithms to solve all generalized coloring problems; in particular, we improved the time complexity known so far for solving Min Cocoloring in cographs. Some attempts have recently been made to handle the generalized coloring problems in cographs

with the additional constraint that, given a cograph $G = (V, E)$, a subset $V' \subseteq V$ is precolored with p cliques and k stable sets, and that this precoloring has to be respected in any (p', k')-coloring of V, where $p' \geq p$ and $k' \geq k$ [38].

In Chapter 4, we tackled the problem of recognizing polar graphs which can also be seen as a generalized coloring problem. We could determine the first class of graphs, namely cographs, where the polar graphs can be recognized in polynomial time. We hope that this result will stimulate the search of similar results in other classes of graphs. Triangulated graphs would be a first candidate for this future research. P_4-reducible graphs, which contain strictly cographs, can also be considered in order to extend the results of both Chapter 3 and 4.

In Chapter 5, we handled generalized coloring problems in line graphs. This research allowed us to detect several subclasses of line graphs where generalized coloring problems behave differently in terms of $\mathcal{NP}$-hardness. In view of the results obtained in this chapter, the existence of known classes of graphs where Min Split-coloring is polynomially solvable while Min Cocoloring is $\mathcal{NP}$-hard remains as an open question.

In Chapter 6, we studied the approximability behavior of generalized coloring problems. We derived several approximation algorithms with a performance guarantee. We also showed that Min Split-coloring and Min Cocoloring are better approximable than Min Coloring from the differential point of view, whereas they behave similarly when using the standard approximation ratio. One may try to improve these results or to find better approximation algorithms in some particular cases.

The utility of generalized coloring problems we introduced is validated by some applications that we analyzed in Chapter 7. These examples show that, beyond their theoretical interest, generalized coloring problems have the potential of bringing new approaches to existing problems; their capacity of modeling is indeed wider than the one of Min Coloring. We also dealt with several problems arising from these applications such as Min Threshold-coloring that we propose to study in the same spirit as the other generalized coloring problems. In this direction, there are unexplored classes of graphs as for example, line graphs of bipartite graphs or triangulated graphs. In addition, we have presented some combinatorial problems involving permutations; these can be studied independently from graph theory.

Many open questions were raised during this dissertation; they aim either at tightening the limits between polynomially solvable and $\mathcal{NP}$-hard for some specific problems, or at contributing to the comparison of the difficulties of two generalized coloring problems.

Coloring and partitioning problems remain a very active field of research in the area of combinatorial optimization. There are many practical motivations to generalize the basic versions of these problems (think of the applications in robotics, telecommunication systems, computer science, etc.). This thesis has explored some of them and there are many research avenues still to be explored.

Bibliography

[1] E. Aarts and J.K. Lenstra, editors. *Local Search in Combinatorial Optimization*. John Wiley & Sons, Inc., New York, NY, USA, 1997.

[2] D. Achlioptas. The complexity of G-free colourability. *Discrete Mathematics*, 165/166:21–30, 1997.

[3] V.E. Alekseev, A. Farrugia, and V.V. Lozin. New results on generalized graph coloring. *Discrete Mathematics and Theoretical Computer Science*, 6(2):215–222, 2004.

[4] L. Alfandari and V.Th. Paschos. Master-slave strategy and polynomial approximation. *Computational Optimization and Applications*, 16:231–245, 2000.

[5] S. Altinakar. Nouvelles bornes supérieures pour le nombre splito-chromatique. Semester project, ROSE, EPFL, 2004.

[6] N. Apollonio. *Some constrained covering problems in graphs*. PhD thesis, University of Roma "La Sapienza", 2002.

[7] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and approximation (Combinatorial optimization problems and their approximability properties)*. Springer-Verlag, 1999.

[8] C. Bazgan, B. Escoffier, and V.Th. Paschos. Poly-APX- and PTAS-completeness in standard and differential approximation. In *Proc. of The 15th Annual International Symposium on Algorithms and Computation, ISAAC 2004*, volume 3341, pages 124–136. LNCS, Springer-Verlag, 2004.

[9] C. Benzaken, P.L. Hammer, and D. de Werra. Split graphs of Dilworth number 2. *Discrete Mathematics*, 55(2):123–127, 1985.

[10] C. Berge. Färbung von Graphen, deren sämtliche bzw. deren ungerade Kreise starr sind. Technical Report Wiss. Zeitung 114, Martin Luther University, Halle-Wittenberg, 1961.

[11] C. Berge. *Graphs and Hypergraphs*. North Holland Publishing Company, 1976.

[12] H.L. Bodlaender. Achromatic number is NP-complete for cographs and interval graphs. *Information Processing Letters*, 31:135–138, 1989.

[13] H.L. Bodlaender and K. Jansen. Restrictions of graph partition problems. *Theoretical Computer Science*, 148:93–109, 1995.

[14] R. Boppana and M.M. Halldórsson. Approximating maximum independent sets by excluding subgraphs. *BIT*, 32:180–196, 1992.

[15] A. Brandstädt. Partitions of graphs into one or two independent sets and cliques. Technical Report N/84/71, Forschungsergebnisse der FSU Jena, 1984. Revised version: Informatik-Berichte FernUniversität / Hagen No. 105, 1/1991, 1991.

[16] A. Brandstädt. Partitions of graphs into one or two independent sets and cliques. *Discrete Mathematics*, 152:47–54, 1996. Corrigendum, *Disc. Math.,* 186:295, 1998.

[17] A. Brandstädt, V. Bang, and J.P. Spinrad. *Graph Classes: a survey*, volume 3 of *SIAM Monographs on Discrete Mathematics and Applications*. SIAM, 1999.

[18] A. Brandstädt and D. Kratsch. On partitions of permutations into increasing and decreasing subsequences. *Journal of Information Processing and Cybernetics*, 22(5/6):263–273, 1986.

[19] A. Brandstädt, V.B. Le, and T. Szymczak. The complexity of some problems related to graph 3-colorability. *Discrete Applied Mathematics*, 89:59–73, 1998.

[20] J.I. Brown. The complexity of generalized graph colorings. *Discrete Applied Mathematics*, 69(3):257–270, 1996.

[21] Zh.A. Chernyak and A.A. Chernyak. About recognizing (α, β)-classes of polar graphs. *Discrete Mathematics*, 62:133–138, 1986.

[22] Zh.A. Chernyak and A.A. Chernyak. Split dimension of graphs. *Discrete Mathematics*, 89:1–6, 1991.

[23] M. Chudnovsky, N. Robertson, P. Seymour, and R. Thomas. The strong perfect graph theorem. *Annals of Mathematics*, 164(1):51–229, 2006.

[24] F. Chung and C.M. Grinstead. A survey of bounds for classical ramsey numbers. *Journal of Graph Theory*, 7:25–37, 1983.

[25] V. Chvátal. Perfectly ordered graphs. In *Topics on Perfect Graphs (C. Berge and V. Chvátal eds.)*, volume 21 of *Annales of Discrete Mathematics*, pages 63–65. 1984.

[26] V. Chvátal and P.L. Hammer. Set-packing and threshold graphs. Technical Report CORR 73-21, Computer Science Department, University of Waterloo, Canada, 1973.

[27] S.A. Cook. The complexity of theorem-proving procedures. In *Proceedings of 3rd Annual ACM Symposium on Theory of Computing*, pages 151–158, New York, 1971. Association for Computing Machinery.

[28] T.H. Cormen, C.S., R.L. Rivest, and C.E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2001.

[29] D.G. Corneil, H. Lerchs, and L. Stewart Burlingham. Complement reducible graphs. *Discrete Applied Mathematics*, 3:163–174, 1981.

[30] D.G. Corneil, Y. Perl, and L.K. Stewart. A linear recognition algorithm for cographs. *SIAM Journal on Computing*, 14(4):926–934, 1985.

[31] G. Cornuéjols and W. Pulleyblank. A matching problem with side conditions. *Discrete Mathematics*, 29:135–159, 1980.

[32] B. Courcelle. Monadic second-order definable graph transductions: a survey. *Theoretical Computer Science*, 126:53–75, 1994.

[33] B. Courcelle, J.A. Makowsky, and U. Rotics. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory of Computing Systems*, 33(2):125–150, 2000.

[34] P. Damaschke. Induced subgraphs and well-quasi-ordering. *Journal of Graph Theory*, 14:427–435, 1990.

[35] D. de Werra. On line perfect graphs. *Mathematical Programming*, 15:236–238, 1978.

[36] D. de Werra, M. Demange, J. Monnot, and V. Paschos. A hypocoloring model for batch scheduling. *Discrete Applied Mathematics*, 146:3–26, 2005.

[37] M. Demange. *Approximation polynomiale de problèmes NP-Complets et programmation linéaire : une nouvelle mesure d'approximation et algorithmes*. PhD thesis, Université de Paris I - Sorbonne, 1994.

[38] M. Demange and T. Ekim. Precoloring with cliques and stable sets in cographs. Technical Report ORWP 06/07, EPFL, 2006.

[39] M. Demange, T. Ekim, and D. de Werra. On the approximation of Min Split-coloring and Min Cocoloring. *Journal of Graph Algorithms and Applications*. to appear.

[40] M. Demange, T. Ekim, and D. de Werra. Partitioning cographs into cliques and stable sets. *Discrete Optimization*, 2:145–153, 2005.

[41] M. Demange, T. Ekim, and D. de Werra. (p, k)-coloring problems in line graphs. *Theoretical Computer Science*, 349(3):462–474, 2005.

[42] M. Demange, T. Ekim, and D. de Werra. Variations of split-coloring in permutation graphs. Technical Report ORWP 06/05, EPFL, 2006. Submitted for publication.

[43] M. Demange, P. Grisoni, and V.Th. Paschos. Differential approximation algorithms for some combinatorial optimization problems. *Theoretical Computer Science*, 209:107–122, 1998.

[44] M. Demange and V.Th. Paschos. On an approximation measure founded on the links between optimization and polynomial approximation theory. *Theoretical Computer Science*, 158:117–141, 1996.

[45] J. Dias. Générateur d'instances pour des problèmes de graphes. Semester project, ROSE, EPFL, 2004.

[46] G.A. Dirac. On rigid circuit graphs. Number 25, pages 71–76. Abh. Math. Sem. Univ. Hamburg, 1961.

[47] R. Duh and M. Fürer. Approximation of k-set cover by semi-local optimization. In *Proc. of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 256–264, 1997.

[48] J. Edmonds. Maximum matching and a polyhedron with 0,1-vertices. *Journal of Research National Bureau of Standards Section B*, 69:125–130, 1965.

[49] J. Edmonds and E.L. Johnson. Matching: A well-solved class of integer linear programs. In R.K. Guy, H. Hanani, N. Sauer, and J. Schönheim, editors, *Proceedings of the Calgary International Conference on Combinatorial Structures and their Applications*, pages 89–92, Gordon and Breach, 1970.

[50] T. Ekim and D. de Werra. On split-coloring problems. *Journal of Combinatorial Optimization*, 10:211–225, 2005. Erratum, 11:125, 2006.

[51] T. Ekim, N.V.R. Mahadev, and D. de Werra. Polar cographs. Technical Report ORWP 05/03, EPFL, 2005. Submitted for publication.

[52] T. Ekim and Th.V. Paschos. Approximation preserving reductions among set covering and vertex covering hierarchies via differential approximation ratio. *International Journal of Computer Mathematics*, 5(81):569 – 582, 2004.

[53] P. Erdös, J. Gimbel, and D. Kratsch. Some extremal results in cochromatic and dichromatic theory. *Journal of Graph Theory*, 15(6):579–585, 1991.

[54] L. Euler. Solutio problematis ad geometriam situs pertinentis. *Opera Omnia*, 7:128–140, 1736.

[55] A. Farrugia. Vertex-partitioning into fixed additive induced-hereditary properties is NP-hard. *Electronic Journal of Combinatorics*, 11(1), 2004.

[56] T. Feder and P. Hell. Matrix partitions of perfect graphs. *Claude Berge Memorial Volume*, 2004.

[57] T. Feder, P. Hell, and W. Hochstättler. Generalized colourings (matrix partitions) of cographs. Manuscript, 2005.

[58] T. Feder, P. Hell, S. Klein, and R. Motwani. List partitions. *SIAM Journal on Discrete Mathematics*, 16(3):449–478, 2003.

[59] T. Feder, P. Hell, S. Klein, L.T. Nogueira, and F. Protti. List matrix partitions of chordal graphs. *Theoretical Computer Science*, 349:52–66, 2005. Latin American Theoretical Information (LATIN 2004), 100-108.

[60] U. Feige, E. Ofek, and U. Wieder. Approximating maximum edge coloring in multigraphs. 108-121. In *APPROX 2002, Rome, Italy*, volume 2462, pages 108 – 121. Springer, 2002.

[61] P. Festa, P.M. Pardalos, and M.G.C. Resende. Feedback set problems. In *Handbook of Combinatorial Optimization*, volume 4. Kluwer Academic Publishers, 1999.

[62] S. Földes and P.L. Hammer. On split graphs and some related questions. In *Problèmes Combinatoires et Théorie des Graphes*, pages 139–140, Orsay, France, 1976. Colloques Internationnaux C.N.R.S. 260.

[63] S. Földes and P.L. Hammer. Split graphs. *Congressus Numerantium*, 19:311–315, 1977.

[64] J. Folkman. Graphs with monochromatic complete subgraphs in every edge coloring. *SIAM Journal of Applied Mathematics*, 18:19–24, 1970.

[65] F.V. Fomin, D. Kratsch, and J.C. Novelli. Approximating minimum cocolorings. *Information Processing Letters*, 84:285–290, 2002.

[66] R.S. Francisco, S. Klein, and L.T. Nogueira. Characterizing (k, l)-cographs. *Electronic notes in Discrete Mathematics*, 22:277–280, 2005. Extended abstract.

[67] D.R. Fulkerson and O.A. Gross. Incidence matrixes and interval graphs. *Pacific Journal of Mathematics*, 15:835–855, 1965.

[68] H.N. Gabow. An efficient reduction technique for degree-constrained subgraph and bidirected network flow problems. In *STOC*, pages 448–456, 1983.

[69] A.V. Gagarin and Y.M. Metelsky. A characterization of $(1, 2)$-polar graphs. *Izvestia Akad. Nauk BSSR, ser. Fiz. Mat. Nauk*, 3:107–112, 1999. (in Russian).

[70] M.R. Garey and D.S. Johnson. *Computers and Intractability - A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, San Francisco, 1979.

[71] M.R. Garey, D.S. Johnson, and L. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1:237–267, 1976.

[72] F. Gavril. Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independant set of a chordal graph. *SIAM Journal on Computing*, 1:180–187, 1972.

[73] F. Gavril. The intersection graphs of subtrees are exactly the chordal graphs. *Journal of Combinatorial Theory, Series B*, 16:47–56, 1974.

[74] F. Gavril. Algorithms for maximum k-colorings and k-coverings of transitive graphs. *Networks*, 17:465–470, 1987.

[75] F. Gilliéron. Collecte d'objets par ordre décroissant de leur taille. Semester project, ROSE, EPFL, 2006.

[76] J. Gimbel. *The chromatic and cochromatic number of a graph.* PhD thesis, Western Michigan University, 1984.

[77] J. Gimbel. Three extremal problems in cochromatic theory. *Rostock. Math. Kolloq.*, 30:73–78, 1986.

[78] J. Gimbel, D. Kratsch, and L. Stewart. On cocolourings and cochromatic numbers of graphs. *Discrete Applied Mathematics*, 48:111–127, 1994.

[79] M.C. Golumbic. Threshold graphs and synchronizing parallel processes. volume 18 of *Combinatorics*, pages 419–428. Colloq. Math. Soc. Janos Bolyai, North Holland, Budapest, 1978.

[80] M.C. Golumbic. *Algorithmic graph theory and perfect graphs.* Computer Science and Applied Mathematics. Academic Press, 1980.

[81] M. Grötschel, L. Lovász, and A. Schrijver. Polynomial algorithms for perfect graphs. *Annales of Discrete Mathematics*, 21:325–356, 1984.

[82] P.L. Hammer and B. Simeone. The splittance of a graph. *Combinatorica*, 1(3):275–284, 1981.

[83] F. Harary. *Graph theory.* Addison-Wesley Publishing Company, 1969.

[84] D. Hartvigsen. *Extensions of Matching Theory.* PhD thesis, Carnegie-Mellon University, 1984.

[85] R. Hassin and S. Khuller. z-approximations. *Journal of Algorithms*, 41:429–442, 2001.

[86] R. Hassin and S. Lahav. Maximizing the number of unused colors in the vertex coloring problem. *Information Processing Letters*, 52:87–90, 1994.

[87] P. Hell, S. Klein, L.T. Nogueira, and F. Protti. Partitioning chordal graphs into independent sets and cliques. *Discrete Applied Mathematics*, 141(1-3):185–194, 2004.

[88] I. Holyer. The NP-completeness of edge-coloring. *Siam Journal on Computing*, 10(4):718–720, 1981.

[89] C.T. Hoàng and V.B. Le. P4-colorings and P4-bipartite graphs. *Discrete Mathematics and Theoretical Computer Science*, 4(2):109–122, 2001.

[90] M. Hujter and Z.S. Tuza. Precoloring extension. II. Graphs classes related to bipartite graphs. *Acta Math. Univ. Comenianae*, 62:1–11, 1993.

[91] K. Jansen. The optimum cost chromatic partition problem. In *CIAC*, pages 25–36, 1997.

[92] K. Jansen, P. Scheffler, and G.J. Woeginger. Maximum covering with d cliques. *Fundamentals of Computing Theory, LNCS*, 710:319–328, 1993.

[93] H.A. Jung. On a class of posets and the corresponding comparability graphs. *Journal of Combinatorial Theory, Series B*, 24(2):125–133, 1978.

[94] G. Karakostas. A better approximation ratio for the Vertex Cover problem. *Electronic Colloquium on Computational Complexity*, (084), 2004.

[95] R.M. Karp. Reducibility among combinatorial problems. In R.E. Miller and J.W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, 1972.

[96] A.E. Kézdy, H.S. Snevily, and C. Wang. Partitioning permutations into increasing and decreasing subsequences. *Journal of Combinatorial Theory, Series A*, 73(2):353–359, 1996.

[97] J. Kratochvíl and I. Schiermeyer. On the computational complexity of (O,P)-partition problems. *Discuss. Math. Graph Theory*, 17:253–258, 1997.

[98] L. Lesniak and J. Straight. The cochromatic number of a graph. *Ars Combinatoria*, 3:34–46, 1977.

[99] L. Lovász. Normal hypergraphs and the perfect graph conjecture. *Discrete Mathematics*, 2(3):253–267, 1972.

[100] N.V.R. Mahadev and U.N. Peled. *Threshold Graphs and Related Topics*, volume 56. Ann. Disc. Mat., North-Holland, 1995.

[101] D.W. Matula. k-components, clusters and slicings in graphs. *Siam Journal of Applied Mathematics*, 22(3):459–480, 1972.

[102] D.W. Matula and L.L. Beck. Smallest-last ordering and clusterings and graph coloring algorithms. *Journal of the Association for Computing Machinery*, 30(3):417–427, 1983.

[103] O. Mel'nikov and P.P. Kozhich. Algorithms for recognizing the polarity of a graph with bounded parameters. *Izvestia Akad. Nauk BSSR, ser. Fiz. Mat. Nauk*, 6:50–54, 1985. (in Russian).

[104] J. Misra and D. Gries. A constructive proof of Vizing's theorem. *Information Processing Letters*, 41:131–133, 1992.

[105] J. Mycielski. Sur le coloriage des graphes. *Colloqium Mathematicum*, 3:161–162, 1955.

[106] C. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.

[107] S. Poljak. A note on the stable sets and coloring of graphs. *Commentationes Mathematicae Universitatis Carolinae*, 15:307–309, 1974.

[108] A. Schrijver. Bipartite edge-colouring in $O(\Delta m)$ time. *Siam Journal on Computing*, 28:841–846, 1999.

[109] H. U. Simon. On approximate solutions for combinatorial optimization problems. *SIAM Journal of Discrete Mathematics*, 3(2):294–310, 1990.

[110] G. Di Stefano, S. Krause, M.E. Lübbecke, and U.T. Zimmermann. On minimum k-modal partitions of permutations. Submitted, October, 2005.

[111] R.E. Tarjan. Depth first search and linear graph algorithms. *SIAM Journal on Computing*, 1:146–160, 1972.

[112] R.E. Tarjan and M. Yannakakis. Addendum: Simple linear time algorithms to test chordality of graphs, test acyclicity of hypergraphs and selectively reduce acyclic hypergraphs. *SIAM Journal on Computing*, 14(1):254–255, 1985.

[113] L.E. Trotter. Line-perfect graphs. *Mathematical Programming*, 12:255–259, 1977.

[114] R.I. Tyshkevich and A.A. Chernyak. Algorithms for the canonical decomposition of a graph and recognizing polarity. *Izvestia Akad. Nauk BSSR, ser. Fiz. Mat. Nauk*, 6:16–23, 1985. (in Russian).

[115] V.V. Vazirani. *Approximation Algorithms.* Springer-Verlag, Berlin, 2001.

[116] V.G. Vizing. On an estimate of the chromatic class of a p-graph. *Diskret. Analiz.*, 3:25–30, 1964. (in Russian).

[117] K. Wagner. Monotonic coverings of finite sets. *Elektron. Inform. Kybernet.*, 20:633–639, 1984.

[118] D.J.A. Welsh and M.B. Powell. An upper bound on the chromatic number of a graph and its application to timetabling problems. *Computer Journal*, 10:85–87, 1967.

[119] M. Yannakakis. Node- and edge-deletion NP-complete problems. Proc. 10th Ann. ACM Symp. on Theory of Computing, pages 253–264, New York, 1978. ACM.

[120] M. Yannakakis and F. Gavril. The maximum k-colorable subgraph problem for chordal graphs. *Information Processing Letters*, 24:133–137, 1987.

[121] E. Zemel. Measuring the quality of approximate solutions to zero-one programming problems. *Mathematics of Operations Research*, 6:319–332, 1981.

[122] N. Zufferey. *Heuristiques pour les problèmes de coloration des sommets d'un graphe et d'affectation de fréquences avec polarités.* PhD thesis, Ecole Polytechnique Fédérale de Lausanne, 2002.

[123] I.E. Zverovich. Perfect cochromatic graphs. Technical Report RRR 16-2000, Rutcor, Rutgers University, 2000.

[124] I.E. Zverovich. r-bounded k-complete bipartite bihypergraphs and generalized split graphs. *Discrete Mathematics*, 247(1-3):261–270, 2002.

Index

Curriculum Vitae

Name	Tınaz Ekim-Aşıcı
Nationality	Turkish
Date of birth	April 29, 1977
Marital Status	Married
Address	Ch. de Bonne Espérance 39
	CH - 1006 Lausanne

Education

2002–2006	PhD student of Prof. Dominique de Werra in Operations research at EPFL, Switzerland
2001–2002	Master degree at Université de Dauphine - Paris IX, France. (DEA de Méthodes Scientifiques de Gestion)
1998–1999	Studies in 3rd year of Mathematics at Université de Technologie et de Sciences de Lille, Bachelor's level degree, France
1996–2001	Studies in Industrial Engineering at Galatasaray University, Industrial Engineering diploma, Turkey
1988–1996	Galatasaray High School (scientific), Turkey

Languages

Turkish	Native language
French	fluently spoken and written
English	fluently spoken and written
Italian	intermediate

Professional Experience

2002–2006	Teaching assistant in ROSE research group at EPFL : Discrete Mathematics, Operations Research, Graphs and Networks, Optimization
2005	Teaching Combinatorial Optimization at Beyrouth University, Lebanon
2005	Teaching Complexity Theory to high school professors of mathematics, Switzerland
2000	Internship in production management at Renault Maïs (2 months), Turkey

Conferences and Talks

2006	*Graphs and Optimization, GO V*, Polar cographs, Leukerbad
2006	*Sixth Czech-Slovak International Symposium on Combinatorics, Graph Theory, Algorithms and Applications*, Polar cographs, Prag
2006	*congrès ROADEF 2006*, On the approximation of Min Split-coloring and Min Cocoloring, Lille
2005	*The Fifth ALIO/EURO Conference on Combinatorial Optimization*, Split-coloring permutation graphs, Paris
2005	*2nd Brazilian Symposium on Graphs, Algorithms, and Combinatorics, GRACO'2005*, (p,k)-coloring line graphs, Angra Dos Reis
2004	*les quatrièmes journées francophones de recherche opérationnelle, FRANCORO IV*, Algorithmes d'approximation et algorithmes séquentiels sur les problèmes de (p,k)-coloration, Fribourg
2004	*Bilkent University*, invited speaker, On generalizations of graph coloring problems, Ankara
2004	*EURO Summer Institute - Optimization and Data Mining*, Partitioning cographs into cliques and stable sets, Ankara
2004	*Mathématiques Discrètes et Sciences Sociales - EHESS*, invited speaker, Problèmes de (p,k)-partitionnement, Paris
2003	*ROADEF Ecole d'Automne*, On Split Coloring Problems, Tours
2003	*First Joint Operations Research Days, IBM Research-The Swiss Operations Research Society*, On Split Coloring Problems, Lausanne
2003	*EURO-INFORMS'03*, Approximation preserving reductions among set covering and vertex covering hierarchies via differential approximation ratio, Istanbul

Additional Publications

- D. de Werra, T. Ekim, C. Raess, Construction of sports schedules with multiple venues, *Discrete Applied Mathematics*, 154:47-58, 2006.
- M. Demange, T. Ekim, D. de Werra, (p,k)-coloring problems in line graphs, Extended Abstract, GRACO 2005, *Electronic Notes in Discrete Mathematics* 19:49-55.
- M. Demange, T. Ekim, D. de Werra, On the approximation of Min Split-coloring and Min Cocoloring, *Annales du Lamsade* No:4-5, Octobre 2005.